Linux Terminal Server Project

Table of Contents

Linux **T**erminal **S**erver **P**roject aide à démarrer les clients LAN à partir d'un seul modèle d'installation qui réside dans une image de machine virtuelle ou un chroot sur le serveur **LTSP**, ou la racine du serveur (/, sans chroot). De cette façon, la maintenance de dizaines ou de centaines de clients sans disque est aussi simple que la maintenance d'un seul PC.

LTSP a été repensé et réécrit à partir de zéro en 2019 par alkisg afin de prendre en charge les nouvelles technologies telles que **systemd**, les environnements de bureau mis à jour, **Wayland**, **UEFI**, etc. Seule la nouvelle version est activement développée, tandis que l'ancienne s'appelle désormais **LTSP5** et est en version minimale.

LTSP configure et utilise automatiquement les outils suivants :

- **iPXE** : chargeur de démarrage réseau qui affiche le menu de démarrage initial du client et charge le noyau/initrd.
- **dnsmasq** ou **isc-dhcp-server**: serveur **DHCP** qui attribue des adresses IP aux clients, ou serveur ProxyDHCP lorsqu'un autre serveur **DHCP** est utilisé, par ex. un routeur.
- **dnsmasq** ou **tftpd-hpa**: serveur **TFTP** qui envoie `ipxe/kernel/initrd` aux clients.
- **dnsmasq**: serveur **DNS** facultatif pour la mise en cache **DNS** ou la liste noire.
- **mksquashfs**: pour créer une copie compressée du modèle **image/chroot/VM** à utiliser comme racine client /.
- **NFS ou NBD**: pour servir l'image **squashfs** au réseau.
- **tmpfs** et **overlayfs**: rendent l'image **squashfs** temporairement inscriptible pour chaque client, comme pour les CD live.
- **NFS** ou **SSHFS**: pour accéder au répertoire `/home` de l'utilisateur qui réside sur le serveur.
- **SSH** ou **SSHFS** ou **LDAP**: pour authentifier les utilisateurs par rapport au serveur.

Installation du serveur

Toutes les commandes de terminal de la documentation doivent être exécutées en tant que root, ce qui signifie qu'il faut d'abord exécuter `sudo -i` sur **Ubuntu** ou `su -` sur **Debian**.

Installation de l'OS du serveur

Le serveur **LTSP** peut être headless, mais il est généralement préférable d'installer le système d'exploitation en utilisant un .iso “de bureau” et non un “serveur”. Tous les environnements de bureau devraient fonctionner correctement, mais MATE reçoit le plus de tests. Toute distribution basée sur .deb qui utilise **systemd** devrait fonctionner ; c'est-à-dire à partir d'**Ubuntu Xenial** et **Debian Jessie** et au-delà. Si on choisit Ubuntu, on peut également envisager de supprimer **Snap** pour éviter certains problèmes.

Installation des packages de serveur LTSP :

```
apt install --install-recommends ltsp ltsp-binaries dnsmasq nfs-kernel-
server openssh-server squashfs-tools ethtool net-tools epoptes
gpasswd -a administrateur epoptes
```

Remplacer administrateur par le nom d'utilisateur de l'administrateur. Si vous n'utilisez pas le PPA, remplacer également ltsp-binaries par ipxe. Description des forfaits susmentionnés :

- **ltsp** : contient le code LTSP, il est commun aux serveurs LTSP et aux clients LTSP.
- **ltsp-binaries**: contient les binaires iPXE et memtest.
- **dnsmasq**: fournit les services TFTP et éventuellement (proxy)DHCP et DNS. Les alternatives possibles sont isc-dhcp-server et tftpd-hpa, mais seul dnsmasq peut faire proxyDHCP, c'est donc la valeur par défaut recommandée.
- **nfs-kernel-server**: exporte l'image disque du client virtuel via NFS.
- **openssh-server**: permet aux clients de s'authentifier et d'accéder à /home via SSHFS.
- **ethtool, net-tools**: permettent de désactiver le contrôle de flux Ethernet, pour améliorer la vitesse du LAN lorsque le serveur est en gigabit et que certains clients sont à 100 Mbps.
- **epoptes**: facultatif ; permet la surveillance du client et le contrôle à distance ; la commande **gpasswd** permet à l'administrateur système d'exécuter **epoptes**.

La liste dess packages nécessaires peut être obtenue avec:
`apt show ltsp | grep ^Suggests`

Configuration du réseau

Il existe deux méthodes courantes pour configurer la mise en réseau **LTSP**. L'une est d'éviter toute configuration ; cela signifie généralement qu'on a une seule carte réseau sur le serveur LTSP et un

serveur DHCP externe, par exemple un routeur, **pfsense**. Dans ce cas, exécuter la commande suivante :

```
ltsp dnsmasq
```

Une autre méthode consiste à avoir un serveur **LTSP** à double carte réseau, où une carte réseau est connectée au réseau normal où se trouve Internet, et l'autre carte réseau est connectée à un commutateur séparé avec uniquement les clients **LTSP**. Pour que cette méthode fonctionne automatiquement, attribuer une adresse IP statique de 192.168.67.1 à la carte réseau interne à l'aide de Network Manager ou de tout autre élément de la distribution, et exécuter :

```
ltsp dnsmasq --proxy-dhcp=0
```

Création des images client

LTSP prend en charge trois méthodes pour cela:

- **Chrootless** (anciennement pnp) : utilise la racine du serveur (/) comme modèle pour les clients. C'est la méthode la plus simple si elle convient car on ne maintient qu'un seul système d'exploitation, et non deux (serveur et image).
- **Image brute de la machine virtuelle**: pour maintenir graphiquement, par ex. une machine virtuelle VirtualBox.
- **Chroot**: pour maintenir manuellement un répertoire chroot à l'aide des commandes de la console.

Dans les cas de machine virtuelle et de chroot, on est censé installer le package ltsp sur l'image, en ajoutant le PPA LTSP et en exécutant `apt install --install-recommends ltsp epoptes-client`, sans spécifier d'autres services. Dans les cas de machine virtuelle et sans chroot, si on utilise des partitions séparées pour certains répertoires comme `/boot` ou `/var`, il faut les inclure. Lorsque l'image est prête, pour l'exporter au format squashfs et la rendre disponible aux clients via NFS, exécuter les commandes suivantes.

Pour sans chroot :

```
ltsp image /
```

Les machines virtuelles doivent être liées symboliquement avant d'exécuter l'image ltsp :

```
ln -s "/home/user/VirtualBox VMs/debian/debian-flat.vmdk"
/srv/ltsp/debian.img
ltsp image debian
```

Pour un chroot dans `/srv/ltsp/my_arc`:

```
ltsp image my_arc
```

Il faut exécuter ces commandes chaque fois qu'on vous installe un nouveau logiciel ou des mises à jour sur l'image et qu'on souhaite exporter sa version mise à jour.

Configurer iPXE

Après avoir créé l'image initiale, ou si vous créez des images supplémentaires, exécutez la commande suivante pour générer un menu **iPXE** et copier les fichiers binaires **iPXE** dans **TFTP** :

```
ltsp ipxe
```

Dans **LTSP5**, **syslinux** était utilisé, mais **iPXE** l'a remplacé car il est beaucoup plus puissant.

Configuration du serveur NFS

Pour configurer le serveur **LTSP** afin qu'il serve les images ou les chroots via **NFS**, exécuter:

```
lsp nfs
```

Générer ltsp.img

Une nouvelle procédure qui n'est pas dans LTSP5 est fourni par la commande suivante :

```
ltsp initrd
```

Cela compresse `/usr/share/ltsp`, `/etc/ltsp`, `/etc/{passwd,group}` et les clés **SSH** publiques du serveur dans `/srv/tftp/ltsp/ltsp.img`, qui est transféré comme “initrd supplémentaire” aux clients lorsqu'ils démarrent. Il faut exécuter `ltsp initrd` après chaque mise à jour du package LTSP, ou lorsqu'on ajoute de nouveaux utilisateurs, ou si on crée ou modifie `/etc/ltsp/ltsp.conf` , qui a remplacé le`lts.conf` dans **LTSP5**.

Configuration des clients

Instructions d'installation de base pour le démarrage réseau des clients Raspberry Pi à partir d'un chroot Raspberry Pi OS (anciennement **Raspbian**) sur un serveur **LTSP**.

Configuration client

La configuration du client est:

- **Pour Pi2**, formater une carte SD avec le système de fichiers fat et y mettre uniquement **bootcode.bin**. Ce fichier se trouve dans `/boot/bootcode.bin` à l'intérieur de l'image Raspberry Pi OS.
- **Pour Pi3B**, démarrer à partir d'une carte SD, exécuter `echo program_usb_boot_mode=1 | sudo tee -a /boot/config.txt` et redémarrer.
- **Le Pi 3B+** prend en charge le **netbooting** prêt à l'emploi.
- **Pour Pi 4 et Pi 400**, démarrer à partir d'une carte SD entièrement mise à jour pour obtenir le dernier micrologiciel, puis exécuter `sudo raspi-config` et sélectionner **Options avancées > Ordre de démarrage > Démarrage réseau**.

Préparation du chroot

Raspberry Pi OS est très optimisé pour les raspberry, c'est donc actuellement une meilleure option que par exemple **Ubuntu MATE** ou d'autres distributions. Le moyen le plus simple de générer un chroot **Raspberry Pi OS** n'est pas avec la commande **debootstrap**, mais en téléchargeant une image. On peut également suivre le guide d'installation de **Raspberry Pi OS** ou utiliser **dd** pour lire la carte SD à partir d'une installation existante de Raspberry Pi ; pour obterir un fichier `raspios.img` non compressé sur le serveur **LTSP**.

```
losetup -rP /dev/loop8 2020-12-02-raspios-buster-armhf-full.img
mount -o ro /dev/loop8p2 /mnt
time cp -a /mnt/. /srv/ltsp/raspios
umount /mnt
mount -o ro /dev/loop8p1 /mnt
cp -a /mnt/. /srv/ltsp/raspios/boot/
umount /mnt
losetup -d /dev/loop8
```

À ce stade, **Raspberry Pi OS** devrait se trouver dans `/srv/ltsp/raspios`. Ce chroot n'est pas encore prêt pour le netbooting, les commandes suivantes sont nécessaires :

```
# Aller au chroot afin d'utiliser les répertoires relatifs
cd /srv/ltsp/raspios
# Masquer les services que nous ne voulons pas dans le netbooting
systemctl mask --root=. dhcpcd dphys-swapfile raspi-config resize2fs_once
# Supprimer les entrées de la carte SD de fstab
echo 'proc            /proc           proc    defaults          0       0'
>./etc/fstab
# Utilise une ligne de commande appropriée pour le démarrage réseau NFS_RW
echo 'ip=dhcp root=/dev/nfs rw
nfsroot=192.168.67.1:/srv/ltsp/raspios,vers=3,tcp,nolock
console=serial0,115200 console=tty1 elevator=deadline fsck.repair=yes
rootwait quiet splash plymouth.ignore-serial-consoles
```

```
modprobe.blacklist=bcm2835_v4l2' >./boot/cmdline.txt
```

Préparation du serveur

Dans `ltsp.conf`, définir **RPI_IMAGE** sur le nom du chroot. Cela sera utilisé par le noyau ltsp pour générer les liens symboliques appropriés de `/srv/tftp/*` vers `/srv/ltsp/raspios/boot/*`.

```
[server]
RPI_IMAGE="raspios"
```

Ensuite, lancer :

```
ltsp kernel raspios
ltsp initrd
ltsp nfs
```

Démarrage réseau NFS_RW

À ce stade, on est prêt à démarrer un seul client en mode **NFS_RW**. Cela signifie que toutes les modifications qu'on effectue sur ce client, comme l'installation de nouveaux programmes, sont directement enregistrées dans `/srv/ltsp/raspios`.

Tout d'abord, exporter le chroot en mode lecture-écriture **NFS** :

```
echo '/srv/ltsp/raspios
*(rw,async,crossmnt,no_subtree_check,no_root_squash,insecure)'
>/etc/exports.d/ltsp-raspios.exports
exportfs -ra
```

On peut remplacer * par une adresse IP pour n'autoriser l'accès qu'à un seul client, ou on peut supprimer le fichier `/etc/exports.d/ltsp-raspios.exports` lorsqu'on a terminé, afin qu'il n'y ait pas de problèmes de sécurité.

Démarrer maintenant un seul client, ajouter le PPA LTSP aux sources et installer les packages côté client :

```
apt install --install-recommends ltsp epoptes-client
```

Netboot en mode LTSP

À ce stade, le chroot contient le code **LTSP** et est prêt à être démarré sur le réseau. Mais il a besoin d'une ligne de commande de noyau différente de celle du mode **NFS_RW**, alors exécuter les

commandes suivantes :

```
echo 'ip=dhcp root=/dev/nfs
nfsroot=192.168.67.1:/srv/ltsp/raspios,vers=3,tcp,nolock
init=/usr/share/ltsp/client/init/init ltsp.image=images/raspios.img
console=serial0,115200 console=tty1 elevator=deadline fsck.repair=yes
rootwait quiet splash plymouth.ignore-serial-consoles
modprobe.blacklist=bcm2835_v4l2' >/srv/ltsp/raspios/boot/cmdline.txt

# Enfin, créer l'image squashfs
ltsp image raspios --mksquashfs-params='-comp lzo'
```

Voilà, on doit maintenant pouvoir démarrer en réseau tous les clients Raspberry Pi.