Création d'un conteneur Turnkey LXC Ubuntu Xfce

Table of Contents

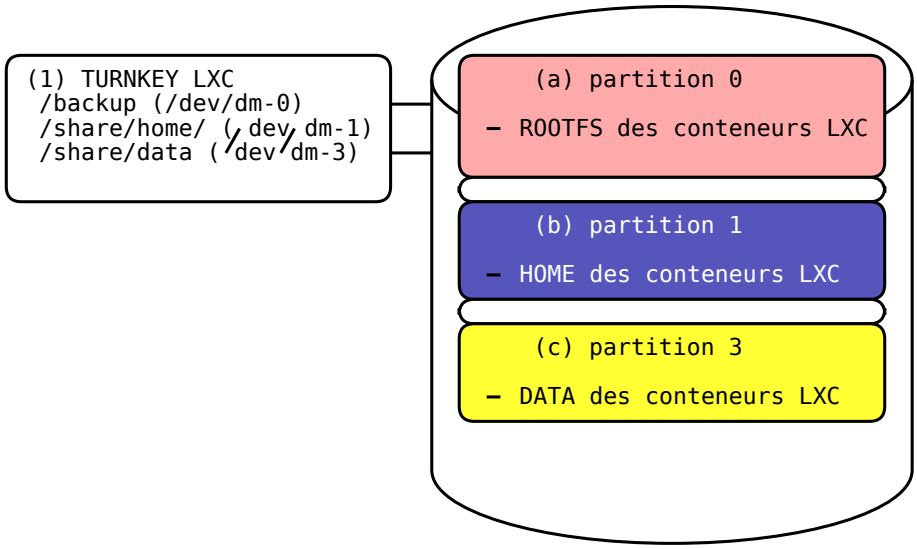
Cette section présente l'installation et la configuration d'une VM ubuntu 18.04 XFCE dans un conteneur LXC sous turnkey

L'ordinateur n'a aucun système installé (et n'en a pas besoins), son disque dur est partitionné de la manière suivante:

- partition 0 btrfs
- partition 1 btrfs
- partition 3 btrfs

Le BIOS de l'ordinateur est configuré pour démarrer sur la clef USB TURNKEY-LXC :

1. L'OS embarqué est un DEBIAN
2. LXC monte les systèmes de fichiers ROOTFS,HOME et DATA
3. Lorsqu'on démarre un conteneur:
    1. son ROOTFS est dans un sous répertoire de la partition 0 montée sur /backup
    2. son HOME est dans un sous répertoire de la partition 1 montée sur /share/home
    3. son DATA est dans un sous répertoire de la partition 3 montée sur /share/data



Mappage des partitions DD et du système de fichiers de la TURNKEY

Configuration de l'hôte

Partage des volumes externes

Pour pouvoir monter automatiquement des volumes externes dans le conteneur il faut les monter dans le système de fichier de TURNKEY-LXC en ajoutant les commandes dans le fichier `/etc/fstab` :

```
echo "/dev/dm-0 /backup  btrfs defaults 0 0" >> /etc/fstab
echo "/dev/dm-1 /share/home/ btrfs defaults,subvol=@home 0 0" >> /etc/fstab
echo "/dev/dm-3 /share/data/ btrfs defaults 0 0" >> /etc/fstab
```

Le fichier `/etc/fstab` de l'hôte doit ressembler à ceci:

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sdb2 during installation
UUID=4637cc09-4076-479f-9ffd-38fade38e1b1 /               ext4
errors=remount-ro 0       1
/dev/mapper/vg01-sp01 none            swap    sw              0       0
/dev/dm-0 /backup  btrfs defaults 0 0
/dev/dm-1 /share/home/ btrfs defaults,subvol=@home 0 0
/dev/dm-3 /share/data/ btrfs defaults 0 0
```

Installation des drivers

Pour installer les drivers **non-free** il faut activer la source des paquets dans `/etc/apt/sources.list.d/sources.list`:

```
sed -i "s/main/main\ non-free/g" /etc/apt/sources.list.d/sources.list
apt update
```

```
apt install firmware-realtek
apt install tklpatch
apt install btrfs-tools
apt install console-data
apt install alsa-utils
```

Il faut récupérer les IDs mineurs et majeurs de chaque périphérique qui devra être utilisé dans la VM, par exemple:

1 - pour récupérer les IDs du périphérique audio la commande `ls -l /dev/snd` renvoi:

```
total 0
drwxr-xr-x 2 root root 80 Dec 17 15:41 by-path
crw-rw---- 1 root audio 116, 7 Dec 17 15:41 controlC0
crw-rw---- 1 root audio 116, 2 Dec 17 15:41 controlC1
crw-rw---- 1 root audio 116, 9 Dec 17 15:41 hwC0D0
crw-rw---- 1 root audio 116, 6 Dec 17 15:41 hwC1D0
...
```

le numéro 116 représente l'iD Majeur du driver audio.

2 - pour récupérer les IDs du périphérique vidéo la commande `ls -l /dev/dri/` renvoi:

```
total 0
drwxr-xr-x 2 root root 80 Aug 31 10:05 by-path
crw-rw---- 1 root video 226, 0 Aug 31 10:05 card0
crw-rw---- 1 root render 226, 128 Aug 31 10:05 renderD128
```

Les numéros 226,0 représentent les IDs majeurs et mineurs de driver vidéo
Les numéros 226,128 représentent les IDs majeurs et mineurs de driver de rendu

3- pour récupérer les IDs pour un périphérique USB, il faut localiser l'appareil en utilisant **lsusb** (Le paquet usbutils doit être installé) pour trouver les périphériques USB connectés.

```
Bus 004 Device 002: ID 8087:8000 Intel Corp.
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 002: ID 8087:8008 Intel Corp.
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 030: ID 046d:c077 Logitech, Inc. M105 Optical Mouse
Bus 001 Device 002: ID 05dc:b055 Lexar Media, Inn
```

Pour la clé USB **Lexar Media, Inc** les numéros de bus **001** et de périphériques **002** permettent d'explorer le chemin de périphérique sous **/dev**, pour connaître les nombres majeurs et mineurs de l'appareil:

```
ls -al /dev/bus/usb/001/002
crw-rw-r-- 1 root root 189, 1 Feb 15 07:53 /dev/bus/usb/001/002
```

Les numéros majeurs et mineurs sont respectivement 189 et 1.

Creation de la VM

Création d'une VM depuis un modèle

Pour créer un conteneur Linux. La première commande est invariablement:

```
sudo lxc-create -t debian -n p1
```

C'est généralement «ubuntu» pour le nom du modèle, mais on peut le remplacer par debian. Le résultat final est le même, il commence à télécharger les composants via le modèle lxc-ubuntu écrit dans `/usr/share/lxc/templates`.

```
lxc-create -t download -n xfce -- -d ubuntu -r bionic -a amd64
```

```
debootstrap is /usr/sbin/debootstrap
Checking cache download in /var/cache/lxc/ubuntu/rootfs-bionic-amd64 ...
Downloading ubuntu minimal ...
I: Retrieving Release
```

Création d'un conteneur à partir d'un ISO

Lorqu'on dispose d'un fichier image local ISO, il existe un moyen d'utiliser à la place cette image .iso locale au lieu d'aller sur le Web.

l'argument fourni à -t est un fichier dans `/usr/share/lxc/templates`. En regardant le modèle lxc-debian, la routine qui effectue le téléchargement s'appelle **download_debian()**, et le travail est effectué par debootstrap:

```
    debootstrap --verbose --variant=minbase --arch=$arch \
    --include=$packages \
    "$release" "$cache/partial-$release-$arch" $MIRROR
```

En regardant la page de manuel, debootstrap peut utiliser un répertoire local pour les fichiers miroir plutôt qu'une adresse réseau

```
…MIRROR can be an http:// or https://URL, a file:///URL or an ssh:///URL.
```

Donc, pour utiliser les données locales:

1. monter l'ISO sur le système de fichiers quelque part;
2. définir la variable d'environnement **MIRROR**;
3. invoquer `lxc-create`.

Si le fichier de version signé n'est pas dans l'ISO il faut passer **–no-check-gpg** à debootstrap, en modifiant le fichier de modèle dans `/usr/share/lxc/templates` pour ajouter cet argument:

```
--- lxc-debian~ 2015-03-04 10:04:12.628619962 +0000
+++ lxc-debian  2015-03-04 10:04:17.420619851 +0000
@@ -232,7 +232,6 @@
     # download a mini debian into a cache
     echo "Downloading debian minimal ..."
     debootstrap --verbose --variant=minbase --arch=$arch \
+   --no-check-gpg \
         --include=$packages \
         "$release" "$cache/partial-$release-$arch" $MIRROR
     if [ $? -ne 0 ]; then
```

Donc, une fois ajusté:

```
# mount -o loop debian-7.8.0-amd64-CD-1.iso /mnt
# export MIRROR=file:///mnt
# lxc-create -t debian -n p1 - -r wheezy
```

Création des users

```
lxc-execute -n xfce -- sudo adduser jacques.nougat --force-badname
lxc-execute -n xfce -- passwd root
```

l'utilsateur **root** est créé par défaut mais il faut lui définir un mot de passe pour pouvoir ce connecter.

Opérations de base

Démarrer/arrêter une VM

```
lxc-stop -n xfce
lxc-start -n xfce
```

Se connecter à la console

```
lxc-console -n xfce
```

Prendre un snapshot

```
lxc-snapshot -n xfce -c /backup/xfce-config.txt -o /backup/xfce-snapshot.txt
```

Configuration de la VM

Configuration dans l'hôte

Fichier de configuration de la VM **xfce**: `/backup/lxc/xfce/config`

```
# Template used to create this container: /usr/share/lxc/templates/lxc-
download
# Parameters passed to the template: -d ubuntu -r bionic -a amd64
# Template script checksum (SHA-1): 740c51206e35463362b735e68b867876048a8baf
# For additional config options, please look at lxc.container.conf(5)

# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (Be aware this has security implications)

# use nat bridge by default

# Distribution configuration
lxc.include = /usr/share/lxc/config/ubuntu.common.conf
lxc.arch = linux64

# Container specific configuration
lxc.include = /etc/lxc/natbridge.conf
lxc.rootfs = /backup/lxc/xfce/rootfs
lxc.rootfs.backend = dir
lxc.utsname = xfce
lxc.cgroup.devices.allow = c 116:* rwm
lxc.cgroup.devices.allow = c 226:0 rwm
lxc.cgroup.devices.allow = c 226:128 rwm
lxc.cgroup.devices.allow = c 4:7 rwm
lxc.cgroup.devices.allow = c 29:0 rwm
lxc.cgroup.devices.allow = c 13:* rwm
lxc.mount.entry = /dev/dri/card0 dev/dri/card0 none
bind,optional,create=file
lxc.mount.entry = /dev/tty7 dev/tty7 none bind,optional,create=file
lxc.mount.entry = /dev/fb0 dev/fb0 none bind,optional,create=file
lxc.mount.entry = /dev/input dev/input none bind,optional,create=dir
lxc.mount.entry = /dev/snd dev/snd none bind,optional,create=dir
lxc.mount.entry = /share/home/jacques.nougat home/jacques.nougat none
bind,rw 0 0
lxc.mount.entry = /share/data data none bind,rw,create=dir 0  0
```

Autorisation passthrough Matériel

Il n'y a pas de relais matériel dans les conteneurs: les périphériques USB, les périphériques PCI, le KVM, etc. sont disponibles que si les autorisations de les utiliser sont définies dans les conteneurs. On peut le faire via cgroup, plus spécifiquement l'option `lxc.cgroup.device.allow`. Pour utiliser cette option, il faut connaître les numéos majeurs et mineurs de l'appareil.

Pour chaque matériel il faut ajouter dans le fichier `/backup/lxc/xfce/config` une ligne donnant les droits rwm (lecture, écriture, montage):

Device	Libellé	Major	Minor
/dev/dri/card0	vidéo	226	0
/dev/dri/renderD128	rendu	226	128
/dev/fb0	framebuffer	29	0
/dev/tty7	console	4	7
/dev/input/*	input	13	*
/dev/bus/usb/001/	usb	189	*

```
lxc.cgroup.devices.allow = c 116:* rwm
lxc.cgroup.devices.allow = c 189:* rwm
lxc.cgroup.devices.allow = c 226:0 rwm
lxc.cgroup.devices.allow = c 226:128 rwm
lxc.cgroup.devices.allow = c 4:7 rwm
lxc.cgroup.devices.allow = c 29:0 rwm
lxc.cgroup.devices.allow = c 13:* rwm
```

L'astérisque qui suit l'iD majeur **189**, qui signifie **tous les périphériques ayant cet ID**

Partage des devices

La commande **lxc.mount.entry** permet de monter la représentation du fichier de l'appareil dans l'espace conteneur. On peut monter un unique périphérique en indiquant le chemin exact, cependant, il peut être judicieux de mapper tout le répertoire (avec tous ses frères et sœurs) au conteneur car il est plus que probable que le nom du fichier change.

Liste des devices (`/dev`) ajoutés dans le fichier `/backup/lxc/xfce/config`:

Périphérique	Libellé
/dev/dri/card0	carte vidéo
/dev/tty7	console système
/dev/fb0	framebuffer
/dev/input	usb et pointeurs de souris
/dev/snd	audio
/dev/bus/usb/001	clefs usb

```
lxc.mount.entry = /dev/dri/card0 dev/dri/card0 none
bind,optional,create=file
lxc.mount.entry = /dev/tty7 dev/tty7 none bind,optional,create=file
lxc.mount.entry = /dev/fb0 dev/fb0 none bind,optional,create=file
lxc.mount.entry = /dev/input dev/input none bind,optional,create=dir
lxc.mount.entry = /dev/snd dev/snd none bind,optional,create=dir
lxc.mount.entry = /dev/bus/usb/001 dev/bus/usb/001 none
```

```
bind,optional,create=dir
```

le chemin de montage cible ne commence pas par une barre oblique! c'est dev pas /dev

Automount USB

Service de montage automatique USB sans utiliser d'outil/service externe, avec uniquement udev et systemd.

Le but du déclencheur udevadm est de dire au noyau d'envoyer des événements pour tous les périphériques présents. Il le fait en écrivant dans `/sys/devices/\*/\*/uevent`. Cela nécessite que sysfs soit monté en lecture-écriture sur /sys (condition **ConditionPathIsReadWrite=/sys** dans le fichier `/lib/systemd/system/udev.service`) en utilisant la commande **lxc.mount.auto** dans le fichier de configuration de la VM:

```
lxc.mount.auto = proc:rw sys:rw
```

La première étape est un bref script shell (`/usr/local/bin/usb-mount.sh` doit avoir les droits d'exécution), qui sera appelé par systemd et s'occupe de créer et de supprimer des points de montage, ainsi que de monter et de démonter les disques:

```
cat <<'EOF' >/usr/local/bin/usb-mount.sh
#!/bin/bash

ACTION=$1
DEVBASE=$2
DEVICE="/dev/${DEVBASE}"

# See if this drive is already mounted
MOUNT_POINT=$(/bin/mount | /bin/grep ${DEVICE} | /usr/bin/awk '{ print $3
}')

do_mount()
{
    if [[ -n ${MOUNT_POINT} ]]; then
        # Already mounted, exit
        exit 1
    fi
    # Get info for this drive: $ID_FS_LABEL, $ID_FS_UUID, and $ID_FS_TYPE
    eval $(/sbin/blkid -o udev ${DEVICE})

    # Figure out a mount point to use
```

```
    LABEL=${ID_FS_LABEL}
    if [[ -z "${LABEL}" ]]; then
        LABEL=${DEVBASE}
    elif /bin/grep -q " /media/${LABEL} " /etc/mtab; then
        # Already in use, make a unique one
        LABEL+="-${DEVBASE}"
    fi
    MOUNT_POINT="/media/${LABEL}"

    /bin/mkdir -p ${MOUNT_POINT}

    # Global mount options
    OPTS="rw,relatime"

    # File system type specific mount options
    if [[ ${ID_FS_TYPE} == "vfat" ]]; then
        OPTS+=",users,gid=100,umask=000,shortname=mixed,utf8=1,flush"
    fi

    if ! /bin/mount -o ${OPTS} ${DEVICE} ${MOUNT_POINT}; then
        # Error during mount process: cleanup mountpoint
        /bin/rmdir ${MOUNT_POINT}
        exit 1
    fi
    # Bonus track: send desktop notification to user
    sudo -u andrea DISPLAY=:0
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus notify-send "Device
${DEVICE} mounted at ${MOUNT_POINT}"
}

do_unmount()
{
    if [[ -n ${MOUNT_POINT} ]]; then
        /bin/umount -l ${DEVICE}
    fi

    # Delete all empty dirs in /media that aren't being used as mount
points.
    for f in /media/* ; do
        if [[ -n $(/usr/bin/find "$f" -maxdepth 0 -type d -empty) ]]; then
            if ! /bin/grep -q " $f " /etc/mtab; then
                /bin/rmdir "$f"
            fi
        fi
    done
}
case "${ACTION}" in
    add)
        do_mount
        ;;
```

```
    remove)
        do_unmount
        ;;
esac

EOF
```

Afin de permettre à systemd d'appeler ce script, créer un fichier d'unité /etc/systemd/system/usb-mount@.service (**«@»** dans la syntaxe du nom de fichier permet de passer le nom du périphérique comme argument).

```
cat <<'EOF' >/etc/systemd/system/usb-mount@.service
[Unit]
Description=Mount USB Drive on %i

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/usr/local/bin/usb-mount.sh add %i
ExecStop=/usr/local/bin/usb-mount.sh remove %i

EOF
```

Enfin, créer un nouveau fichier dans /etc/udev/rules.d/99-local.rules contenant quelques règles udev pour démarrer et arrêter le service d'unité systemd sur hotplug / unplug :

```
cat <<'EOF' >/etc/udev/rules.d/99-local.rules
KERNEL=="sd[a-z][0-9]", SUBSYSTEMS=="usb", ACTION=="add",
RUN+="/bin/systemctl start usb-mount@%k.service"
KERNEL=="sd[a-z][0-9]", SUBSYSTEMS=="usb", ACTION=="remove",
RUN+="/bin/systemctl stop usb-mount@%k.service"

EOF
```

Donner les droits d'exécution au script-shell, puis prendre en compte le service et le fichier de règles udev:

```
chmod +x /usr/local/bin/usb-mount.sh
udevadm control --reload-rules
systemctl daemon-reload
```

Partage des dossiers

Ajouter dans le fichier `/backup/lxc/xfce/config` les points de montage des dossiers partagés

```
lxc.mount.entry = /share/home/jacques.nougat home/jacques.nougat none
```

```
bind,rw 0 0
lxc.mount.entry = /share/data data none bind,rw,create=dir 0  0
lxc.mount.entry = /media media none bind,rw,create=dir 0  0
```

l'absence de / devant les points de montage, car on indique ici un chemin relatif dont la racine est le dossier ou est stocké le système de fichier du conteneur.

Configuration du proxy iptables

Turnkey LXC expose facilement les services de conteneurs NAT au travers :

Pour exposer les services du conteneur au réseau on peut utilise iptables sur l'hôte pour transférer le trafic qu'il reçoit sur le port X vers le conteneur sur le port Y.

Dans l'hôte relever l'adresse ip du conteneur :

```
lxc-info -n xfce -iH

192.168.121.187
```

LXC attribue des adresses ip par DHCP au conteneur, alors qu'iptables lie un flux à une adresse unique. Pour attribuer une adresse statique à la VM, voir la section Configuration du réseau

Le script **iptables-nat** permet de configurer facilement le pare-feu iptables:

```
Syntax: iptables-nat action s_port d_addr:d_port
Add or delete iptables nat configurations

Arguments::

    action          action to perform (add|del|info)
    s_port          source port on host
    d_addr:d_port   destination ip address and port

Examples::

    iptables-nat add 2222 192.168.121.150:22
    iptables-nat del 2222 192.168.121.150:22
```

Pour la VM xfce4 on a défini les règles suivntes

port de l'hôte	port dans la VM	rôle	iptables-nat
5900	5900	VNC	iptables-nat add 5900 192.168.121.87:5900
5901	5901	VNC	iptables-nat add 5901 192.168.121.87:5901
3389	3389	RDP	iptables-nat add 3389 192.168.121.87:3389
80	80	HTTP	iptables-nat add 80 192.168.121.87:80
443	443	HTTPS	iptables-nat add 443 192.168.121.87:443
2222	22	SSH	iptables-nat add 2222 192.168.121.87:22

```
iptables -S

# Network configuration
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-N f2b-sshd
-A INPUT -p tcp -m multiport --dports 22 -j f2b-sshd
-A INPUT -m state --state INVALID -j DROP
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -s xx.xx.xxx.0/24 -i br0 -m conntrack --ctstate NEW -j ACCEPT
-A INPUT -s 192.168.121.0/24 -i natbr0 -m conntrack --ctstate NEW -j ACCEPT
-A INPUT -i natbr0 -p udp -m udp --sport 68 --dport 67 -j ACCEPT
-A FORWARD -d 192.168.121.87/32 -p tcp -m state --state NEW -m tcp --dport
22 -j ACCEPT
-A FORWARD -d 192.168.121.87/32 -p tcp -m state --state NEW -m tcp --dport
443 -j ACCEPT
-A FORWARD -d 192.168.121.87/32 -p tcp -m state --state NEW -m tcp --dport
80 -j ACCEPT
-A FORWARD -d 192.168.121.87/32 -p tcp -m state --state NEW -m tcp --dport
3389 -j ACCEPT
-A FORWARD -d 192.168.121.87/32 -p tcp -m state --state NEW -m tcp --dport
5901 -j ACCEPT
-A FORWARD -d 192.168.121.87/32 -p tcp -m state --state NEW -m tcp --dport
5900 -j ACCEPT
-A FORWARD -d 192.168.121.0/24 -p icmp -m icmp --icmp-type 8 -j DROP
-A FORWARD -d 192.168.121.0/24 -p tcp -m conntrack --ctstate NEW -j DROP
-A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s xx.xx.xxx.0/24 -i br0 -m conntrack --ctstate NEW -j ACCEPT
-A FORWARD -s 192.168.121.0/24 -i natbr0 -m conntrack --ctstate NEW -j
ACCEPT
-A f2b-sshd -j RETURN
```

Configuration du réseau

On peut définir l'adresse et la passerelle depuis l'hôte et configurer le conteneur pour ne pas affecter du tout l'interface à l'aide du mot-clé manual.

Ifupdown

Placer ceci dans les fichiers `/etc/network/interfaces` des conteneurs:

```
auto eth0
iface eth0 inet manuel
```

Configurer l'interface dans le fichier de configuration du conteneur

```
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = lxc-bridge-nat
lxc.network.ipv4 = 192.168.100.16/24
lxc.network.ipv4.gateway = auto
```

L'invité se comportera comme si son BIOS avait déjà configuré l'interface et l'utilisera simplement.

Netplan

Dans Ubuntu 18.04, l'ancien **ifupdown** est considéré comme obsolète. Et il faut migrer vers **systemd-networkd + netplan**

Remplacer le contenu du fichier sudo vi `/etc/netplan/10-lxc.yaml`

```
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: true
      dhcp-identifier: mac
```

par celui ci:

```
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: no
      dhcp6: no
      addresses: [ 192.168.121.87/24 ]
      gateway4: 192.168.121.1
      nameservers:
        addresses: [10.154.59.104,10.156.32.33]
```

lxc-snapshot -n xfce -c /backup/xfce-config.txt -o /backup/xfce-snapshot.txt

Configuration dans la VM

Configuration des locales

```
locale-gen fr-FR
locale-gen fr_FR.UTF-8
apt update-locale
```

Modification des droits utilisateurs

```
adduser jacques.nougat sudo
addgroup admin
adduser jacques.nougat admin
```

Installation des paquets

```
export https_proxy="http://proxy.infra.dgfip:3128"
export httpt_proxy="http://proxy.infra.dgfip:3128"
apt install xfce4 thunar-archive-plugin cups system-config-printer-gnome
evince
apt install libreoffice libreoffice-nlpsolver libreoffice-help-fr
libreoffice-l10n-fr
```

Le réglage par défaut de Xorg (“**allowed_users=console**” via Xorg.wrap), est trop restrictif. Cela exige que l'utilisateur soit connecté à la console, et affecte tout logiciel qui souhaite créer un framebuffer virtuel à l'aide du logiciel Xorg. En particulier, pour faire fonctionner xfce4 dans la console -vt7.

Passer à “**allowed_users=anybody**”, en indiquant dans le fichier `/etc/X11/Xwrapper.config`:

```
allowed_users=anybody
needs_root_rights=yes
```

L'ISO d'installation du socle DGFIP fournit les paquets suivants
- **cups-ppd-msnrl_1.0.6_all.deb**: pilote d'impression
- **firefox-esr-msnrl_68.4.1-1_all.deb**: firefox esr
- **msnrl-cisco-anyconnect_4.5.05030_all.deb**: CISCO AnyConnect Secure Mobility Client
- **msnrl-mce_1.1.0_all.deb** courielleur mce thunderbird dépend de gnome-shell
- **msnrl-numlock_0.1.2_all.deb**: Par défaut, sous Ubuntu, le verrouillage du pavé numérique au démarrage n'est pas activé, numlock vient corriger cette situation

- **msttcorefonts-msnrl_1.0-2_all.deb**: Fontes MS TrueType
- **msnrl-keyring_2018.01.22_all.deb**: Clé publique de vérification des paquets empaquetés par la MSNRL
- **virtualbox-extensionpack-msnrl_5.2.18_all.deb**: ExtensionPack Virtualbox

Clavier et souris

Pavé numérique

Il peut arrivé que lorsqu'on appuie sur **VERR NUM**, rien ne se passe, avec **CTRL + ALT + F1** ou **SHIFT + VERR NUM**, celui ci fonctionnera.

Intégration du pointeur dans la config Xfce

Editer le fichier `/usr/share/X11/xorg.conf.d/10-evdev.conf`

```
#
# Catch-all evdev loader for udev-based systems
# We don't simply match on any device since that also adds accelerometers
# and other devices that we don't really want to use. The list below
# matches everything but joysticks.

Section "InputClass"
        Identifier "evdev pointer catchall"
        MatchIsPointer "on"
        MatchDevicePath "/dev/input/event*"
        Driver "evdev"
EndSection

Section "InputClass"
        Identifier "evdev keyboard catchall"
        MatchIsKeyboard "on"
        MatchDevicePath "/dev/input/event*"
        Driver "evdev"
EndSection

Section "InputClass"
        Identifier "evdev touchpad catchall"
        MatchIsTouchpad "on"
        MatchDevicePath "/dev/input/event*"
        Driver "evdev"
EndSection

Section "InputClass"
        Identifier "evdev tablet catchall"
```

```
        MatchIsTablet "on"
        MatchDevicePath "/dev/input/event*"
        Driver "evdev"
EndSection

Section "InputClass"
        Identifier "evdev touchscreen catchall"
        MatchIsTouchscreen "on"
        MatchDevicePath "/dev/input/event*"
        Driver "evdev"
EndSection
```

Imprimante

Installer **cups** et **system-config-printer-gnome**:

1. installer les paquets avec la commande `apt install cups system-config-printer-gnome`
2. Télécharger le fichier PostScript Printer Description ou *.ppd correspondant au modèle d'imprimante utilisé.
3. décompacter l'archive.
4. en **root** lancer **system-config-printer**
5. appuyer sur add pour ajouter une imprimante
6. choisir **Network printer>Find Network Printer**
7. indiquer l'adresse ip de l'imprimante dans **Host**
8. appuyer sur probe
9. prendre le fichier ppd télécharger comme driver

Donner les droits d'utilisation à l'utisateur normal e redémarrer cups (stop+start et non restart):

```
sudo usermod -aG lpadmin jacques.nougat
sudo service cups start
```

Le service **cups.socket** doit être **Active: active (running)**

```
systemctl status cups.socket

Failed to dump process list, ignoring: No such file or directory
  cups.socket - CUPS Scheduler
   Loaded: loaded (/lib/systemd/system/cups.socket; enabled; vendor preset:
enab
   Active: active (running) since Mon 2021-01-04 09:42:49 UTC; 2h 7min ago
   Listen: /run/cups/cups.sock (Stream)
   CGroup: /system.slice/cups.socket
```

Gnome Teminal

Si lors du lancement du terminal Gnome une erreur est retournée

```
$ gnome-terminal
Error constructing proxy for org.gnome.Terminal:
/org/gnome/Terminal/Factory0:
Error calling StartServiceByName for org.gnome.Terminal:
Process /usr/lib/gnome-terminal/gnome-terminal-server exited with status 8
```

Mettre à jour le lanceur du terminal:

1. Clic droit sur l'icone du lanceur de terminal
2. choisir l'item **Properties**
3. cliquer sur Edit
4. Puis remplacer la commande `exo-open --launch TerminalEmulator` par `/usr/bin/dbus-launch /usr/bin/gnome-terminal`

NGINX/PHP

PHP-FPM (FastCGI Process Manager) est une alternative à l'implémentation FastCGI de PHP avec quelques fonctionnalités supplémentaires utiles pour les sites à fort trafic. C'est la méthode préférée de traitement des pages PHP avec NGINX et est plus rapide que les méthodes traditionnelles basées sur CGI telles que SUPHP ou mod_php pour exécuter un script PHP.

```
apt install nginx php-fpm php7.2-sqlite mysql-server php-mysql php-curl php-
ldap
```

Si PHP ne démarre pas correctement, vérifier que le répertoire `/run/php` existe bien, si ce n'est pas ajouter la création du répertoire dans la section start du script init.d de php:

```
sudo vi /etc/init.d/php7.2-fpm
```

```
case "$1" in
    start)
        if init_is_upstart; then
            exit 1
        fi
        [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
        do_tmpfiles $VERBOSE
        mkdir --mode=07500 /run/php
        chown www-data:www-data /run/php
```

redémarrer php

```
sudo systemctl start php7.2-fpm
```

L'extension **PHP pcntl** peut prendre en charge le fonctionnement multi-thread (par exemple pour la gestion de la balise de VLAN du standard IEEE 802.1Q). (Les systèmes non basés sur Unix ne prennent pas en charge ce module). L'installation de cette extension est manuelle et nécessite l'installation des outils de développementPHP:
1) - installer les outils de développement:
```
apt-get install php7.2-dev -y
```
2) - télécharger les sources PHP 7.2.24:
```
cd /tmp
wget http://php.net/distributions/php-7.2.24.tar.gz
```
3) - décompresser le fichier source de PHP 7.2.24:
```
tar zxvf php-7.2.24.tar.gz
```
- compiler le module PCNTL:
```
cd php-7.2.24/ext/pcntl
phpize
./configure
make
```
4) - copier le module PCNTL du répertoire modules vers le répertoire PHP de travail:
```
cd modules
cp pcntl.so /usr/lib/php/20170718
```
5) - créer le fichier pcntl.ini:
```
echo "extension=pcntl.so" > /etc/php/7.2/mods-available/pcntl.ini
```
6) - créer un lien symbolique vers le fichier pcntl.ini créé à l'étape précédente:
```
ln -s /etc/php/7.2/mods-available/pcntl.ini
/etc/php/7.2/fpm/conf.d/20-pcntl.ini
```
7)- modifier la ligne commençant par “disable*functions” en “;disable*functions” dans le fichier php.ini:
```
vi /etc/php/7.2/fpm/php.ini
```
8) - installer le complément PHP-ZIP pour PHP:
```
apt installer zip unzip php7.2-zip
```
9) - redémarrer le service

Gestionnaire de fichier Thunar

Pour afficher les vignettes d'image/vidéo dans Thunar installer les paquets suivants:

```
killall thunar
sudo apt install tumbler tumbler-plugins-extra ffmpegthumbnailer
thunar -q
```

Cela nécessitera de redémarrer la machine.

Capture d'écran

Installer l'application `xfce4-screenshooter`

```
apt install xfce4-screenshooter
```

Dans `Menu XFCE -> Paramètres -> Clavier -> Raccourcis d'application`, ajouter la commande `xfce4-screenshooter -f` pour utiliser la touche “PrintScreen” afin de prendre des captures d'écran en plein écran.

Prise de main a distance

Le VPN0 n'autorise que le protocole RDP® pour prendre un bureau a distance, si l'on souhaite se connecter depuis ces machines sur un serveur Ubuntu, une seule solution : installer XRDP sur ce dernier.

Xrdp est une couche s'installant sur le serveur. Elle fait office de serveur RDP® pour le client qui veut se connecter, et se comporte comme un client VNC pour le serveur ubuntu.

Entre les deux, se glisse un gestionnaire de session dénommé **sesman**, qui :

- donne le choix, à la connexion, entre différents modules. Ceux-ci sont listés dans /etc/xrdp/xrdp.ini, on y trouve bien sûr VNC.
- appelle PAM pour l'authentification.
- fait en sorte que le client VNC, avatar du client RDP, se connecte à une session VNC fonctionnelle. Le plus souvent en la créant, à l'aide des paramètres choisis dans `/etc/xrdp/sesman.ini` et `/etc/xrdp/startwm.sh`.

Xrdp s'interface avec un serveur vnc, il faut Installer un serveur vnc4server ou similaire (par exemple x11vnc)

```
apt install x11vnc xrdp
```

Créer le service x11vnc dans le fichier `/etc/systemd/system/x11vnc.service`

```
# Fichier x11vnc.service pour Debian ou Ubuntu avec systemd
#
[Unit]
Description=VNC Server for X11
After=multi-user.target network.target

[Service]
Restart=always
```

```
ExecStart=/usr/bin/x11vnc -display :0 -forever -loop -noxdamage -repeat -
rfbport 5900 -shared -o /var/log/x11vnc.log
ExecStop=/usr/bin/x11vnc -R stop

[Install]
WantedBy=multi-user.target
```

Exécuter puis activer le service

```
systemctl daemon-reload
systemctl enable x11vnc.service
```

Editer le fichier `/etc/xrdp/xrdp.ini` pour ne conserver que les sections [globals] et [xrdp1] que l'on renomme pour le fun :

```
[globals]
bitmap_cache=yes
bitmap_compression=yes
port=3389
crypt_level=low
channel_code=1

[xrdp1]
name=legaub vous invite sur un serveur Ubuntu
lib=libvnc.so
username=ask
password=ask
ip=127.0.0.1
port=-1
```

Redémarrer avec :

```
sudo /etc/init.d/xrdp restart
```

On peut désormais se connecter au serveur en utilisant le protocole RDP®.

Configuration du clavier: Lorsque les configurations du clavier en local et en distant sont différentes (si Ubuntu est en anglais à la base et même si en local, on a bien configuré le clavier en azerty, le clavier distant reste en qwerty). Voici la commande à exécuter en local, (pas en session distante)

```
sudo xrdp-genkeymap /etc/xrdp/km-0409.ini
```

relancer la session distante

Java JNLP

Le **J**ava **N**etwork **L**aunch **P**rotocol (**JNLP**) est un protocole qui permet de lancer une application sur un bureau client en utilisant des ressources hébergées sur un serveur Web distant. Le logiciel Java Plug-in et le logiciel Java Web Start sont considérés comme des clients **JNLP** car ils peuvent lancer des applets et des applications hébergées à distance sur un bureau client.

```
sudo apt update
sudo apt install default-jdk
sudo apt install icedtea-netx icedtea-plugin
```

Pour les distributions à base de RedHat (CENTOS) installer epel-release java-11-openjdk java-11-openjdk-devel icedtea-web