TüBix 2019: Integration von Linux-Containern auf dem Desktop mit LXC

Inhalt:

- Kurze Worte zu LXC resp. Container i.A.
- Mögliche Vorteile durch Nutzung von Containern auf dem Desktop
- Vorstellung konkreter Szenarios und Illustration an konkreten Beispielen

Was sind (Linux-)Container? Was ist LXC? (Kurzfassung)

- Merkmal Container Virtualisierung: Host- und Gastsystem teilen denselben Kernel
- Linux-Container basieren auf Kernel-Features `cgroups` und `namespaces`
  - `cgroups`: Zuständig für Zugriffsverwaltung auf System- u. Hardware-Ressourcen auf Hostsystem
  - `namespaces`: Container-root != Host-root

- LXC = Frontend für Erstellung / Verwaltung von Linux-Containern
- Weitere Frontends: LXD, Docker, ..

Privilegierter vs. unprivilegierter Container:

- Separate UID/GID auf Host- u. Gastsystem
  - privilegiert: root UID = 0 (Range: 0-65536)
  - unprivilegiert: root UID = 100000 (Range: 100000- 165536)

- LXC-Konfiguration für unprivilegierten Container:

```
# run as unprivileged container
lxc.idmap = u 0 100000 65536
lxc.idmap = g 0 100000 65536
```

- Konfiguration auf Host (in */etc/subuid* und */etc/subgid*):

```
meffisto:100000:900000
```

Prinzipielle Vorteile von Containern (auf dem Desktop):

- Sehr effizient / ressourcenschonend im Vergleich zu Hardware-Virtualisierung (Xen, KVM, VirtualBox, etc.)
- Direkter Zugriff auf Hardware-Ressourcen (z.B. Grafik) via Host-Kernel
- Separation von Programmen / Diensten auf Host und Gastsystem (Container)
- Separate Netzwerkschnittstellen
- Snapshots: Schnelles Erstellen, Wiederherstellen, Clonen von Test- u. Entwicklungsumgebungen
- Systemzustand unabhängig vom Host konservieren (kein Update/Upgrade)
  → alte Software weiternutzen (z.B. Spiele)

LXC Szenarios: 32 Bit (Gast) vs. 64 Bit (Host)

- Reines 64Bit-System (Host), keine Mischung (mixed environment) mit 32 Bit Libs
  → kein `dpkg --add-architecture i386` mehr nötig
- Bessere Kompatibilität zu „echtem“ 32 Bit, weniger Fehler/Probleme im Vergleich zu mixed environment (z.B. bei Spielen)
- Proprietäre 32-Anwendung isolieren (HBCI mit Smartcard und Moneyplex)
  → Optional: Container auf (verschlüsselten) USB-Stick separieren (und wegschließen)
- Systemzustand konservieren (kein Update/Upgrade), alte 32-Bit-Software weiternutzen

LXC Szenarios: Distributionen mischen

- Vorteile von verschiedenen, spezifischen Distributionen nutzen (z.B. spezifische Pakete)
- Best of both worlds: Gentoo Host + Debian Container oder Debian Host + Gentoo Container
- Keine Mischung von stable und testing/unstable auf Hostsystem mehr notwendig
  → testing/unstable im Container einrichten (sofern Host-Kernel neu genug)
- Mischung von alt und neu (z.B. Debian 8, Debian 9, Debian 10)
  → Verschiedene Systeme für verschiedene (alte) Programmversionen (z.B. TeamViewer)
- Hostsystem schlank und sauber halten. Beispiel: Container für Java-Applikationen (OpenJRE vs. Oracle Java)

LXC Beispiel: (Privilegierter) Container für Spiele (32 Bit)

- **Ziel:** Spiele vom Arbeitssystem entkoppeln
- Container / Umgebung spezifisch an das Spiel anpassbar (unabhängig vom Host)
- Aktueller Grafiktreiber (Kernel) vom Hostsystem; Mesa/OpenGL im Container
- LXC-Konfiguration für Desktop-Nutzung:

```
# Sound device nodes
lxc.cgroup.devices.allow = c 116:* rwm
# Video output
lxc.mount.entry = /dev/dri dev/dri none bind,optional,create=dir
# Sound output
lxc.mount.entry = /dev/snd dev/snd none bind,optional,create=dir
# Video Output
lxc.mount.entry = /tmp/.X11-unix tmp/.X11-unix none bind,optional,create=dir
# Pulse Audio socket
lxc.hook.pre-start = /var/lib/lxc/debian9-Games/setup-pulse.sh
```

- Programm ausführen:

```
$ sudo lxc-attach --clear-env -n $CONTAINER -- sudo -u $LXC_USER -i env DISPLAY=:0 \
PULSE_SERVER=/home/meffisto/.pulse_socket quake3 > /dev/null 2>&1
```

LXC Beispiel: Unprivilegierter Container für Webbrowsing

- **Ziel:** Firefox in unprivilegiertem Container ausführen
- Zutaten auf [H]ost resp. [G]ast:
  - H: Xephyr: Separater X-Server (im X-Server des Host); DISPLAY=:11
  - G: evilwm: rudimentärer Window-Manager
  - H + G: PulseAudio: Audioausgabe via Netzwerk zum PA-Server auf Host
  - H: Xclip: Clipboardsynchronisation zwischen Host (DISPLAY=:0) und Gast (DISPLAY=:11)
- Firefox u.a. ausführen:

```
$ lxc-attach --clear-env -n $CONTAINER -- sudo -u $LXC_USER -i env DISPLAY=:11 \
     evilwm -snap 10 -bw 0 &
$ lxc-attach --clear-env -n $CONTAINER -- sudo -u $LXC_USER -i env DISPLAY=:11 \
     PULSE_SERVER=tcp:192.168.0.32:4713 firejail firefox-esr -P --no-remote &
$ xclip -selection clip -o -display :0 | xclip -selection clip -i -display :11
```

- LXC-Konfiguration für Downloads-Verzeichnis (bind-mount ~/Downloads):

```
# Mount partitions
lxc.mount.entry=/home/meffisto/Downloads/ /var/lib/lxc/debian9-webbrowser/rootfs\
/home/mozilla/Downloads none bind,nodev,noexec 0 0
```

- Bonus: Container Clonen, TOR Browser Bundle installieren, fertig

LXC Beispiel: JACK Audio + externes USB Audio Device

- **Ziel:** Ubuntu Studio o.ä.nutzen ohne Reboot
- Keine (konfliktträchtige) Mischung der Soundsysteme: JACK im Container, PA auf dem Host
- Direktzugriff auf USB Audio Device (Input + Output):

```
# USB Audio Interface
lxc.cgroup.devices.allow = c 189:* rwm
lxc.mount.entry = /dev/bus/usb/002 dev/bus/usb/002 none bind,optional,create=dir
```

- Audiobearbeitung benötigt zahlreiche Pakete, Host bleibt hiervon unberührt und schlank
- JACK benötigt zwingend DBUS (läuft getrennt vom Host): `$ dbus-run-session -- qjackctl &`
- Echtzeitkernel nicht notwendig (sofern nicht dutzende von Spuren und Effekten aktiv sind)

LXC Beispiel: WINE

- **Ziel:** Windows-Software nachhaltig unter Linux nutzbar machen
- Kombination verschiedener WINE-Versionen auf unterschiedlichen Distributionen möglich (Kernel bleibt jedoch identisch) – PlayOnLinux enttäuscht hier!
- Snapshot- u. Cloning-Feature von Containern extrem hilfreich, z.B. wenn Registry-Einträge nach WINE-Update ungültig (Software-Lizensierung)
- Zustand konservieren: WINE resp. Container nicht mehr updaten
- Separate Netzwerkschnittstelle jeder WINE-Installation (resp. im jeweiligen Container)
- Externe Verzeichnisse per bind-mount nur read-only in Container einbinden
- Performance optimieren durch Wahl des Laufwerkstyps (SSD, VelociRaptor HDD, HDD, ..) mittels bind-mount Möglichkeit von LXC:
  - Verzeichnis mit Spieldaten auf SSD
  - Verzeichnis mit Videodaten auf HDD
- Performance limitieren: `cgroups` limitieren CPU-Zeit für Container, falls ein Prozess Amok läuft

Anhang:

- Skript für Pulse Audio Socket (*setup-pulse.sh*):

```
#!/bin/sh

PA_USER=meffisto
PULSE_PATH=$LXC_ROOTFS_PATH/home/$PA_USER/.pulse_socket

if [ ! -e "$PULSE_PATH" ] || [ -z "$(lsof -n $PULSE_PATH 2>&1)" ]; then
    sudo -u $PA_USER -i pactl load-module module-native-protocol-unix auth-anonymous=1
socket=$PULSE_PATH
fi
```

- Links für Interessierte:
  - https://linuxcontainers.org/lxc/getting-started/
  - https://wiki.gentoo.org/wiki/LXC#Unprivileged_containers
  - http://www.lug-erding.de/artikel/LXC.html