LXC Container

* USB Device an Container durchreichen
* Festplatte an Container durchreichen
* Docker in einem LXC Container betreiben

LXC Container Template erstellen

1. LXC Container mit integriertem Template – das wird die Basis für das eigene Template sein - erstellen.
2. Container starten
3. Container betreten Entweder durch die KVM Konsole oder per SSH auf dem Proxmox Host **pct enter [ContainerID]**
4. Alles einstellen, was man möchte, z.B. eigene bashrc, User, SSH Keys, Applikationen installieren...
5. Container aufräumen

```
apt clean
apt autoclean
apt autoremove
rm /etc/resolv.conf
rm /etc/hostname
```

Puppet Agent Zertifikat bereinigen

```
On the master:
  puppet cert clean t-template.tuxnet.lan
On the agent:
  1a. On most platforms: find /var/lib/puppet/ssl -name t-
template.tuxnet.lan.pem -delete
  1b. On Windows: del "\var\lib\puppet\ssl\certs\t-template.tuxnet.lan.pem"
/f
  2. puppet agent -t
```

1. Container verlassen
2. In der Proxmox Webgui alle Netzwerkinterfaces des Containers entfernen
3. Über die Webgui ein Backup vom Container erstellen
   Mode: Stop
   Compression: Gzip
   Die Compression muss Gzip sein weil die Proxmox Webgui das Template später sonst nicht anzeigt
4. Per SSH auf dem Proxmox Host in den allgemeinen Container Backupordner wechseln
   Standardmäßig **/var/lib/vz/dump/**
   In dem Ordner liegt jetzt das Backup als .tar.gz mit .log File
5. Das .tar.gz File muss in den richtigen Ordner für Container Templates geschoben werden
   Standardmäßig **/var/lib/vz/template/cache/[Templatename].tar.gz**
   Der Templatename kann frei vergeben werden. Was hier eingetragen wird, wird später in der

Webgui angezeigt. Ich benenne es immer nach dem Schema [distribution-version_templatedatum_architecture].tar.gz Also z.B. **ubuntu-16.04-standard_2017-05-17_amd64.tar.gz**

Und wie macht man nachträglich Änderungen am eigenen Template? Einfach den Template Container starten und alle Schritte, mit Ausnahme des ersten, der Reihe nach wiederholen.

Alternative feine Variante

Der neue Container wird noch nicht gestartet!

Konfiguration auf dem Host

ssh

Erfahrungsgemäss wird Port 22 reativ oft angegriffen. Deshalb ändere ich den SSH-Port auf all meinen Sytemen auf Port 55.

/Pool1/subvol-250-disk-1/etc/ssh/sshd.conf Wir suchen nach „port 22“ und ersetzen das mit „port 55“ (Backup gemacht?).

Auch „PermitRootLogin“ auf „yes“ setzen. In den CT macht ein „no“ keinen Sinn, denn ich habe keine andern Benutzer aktiv!

Auf dem Host ist auch noch Port 22 aktiv. Also die obigen Anpassungen auch noch „/etc/ssh/sshd.conf“ nachtragen, wobei hier ein „no“ angebracht ist.

hosts / resolv Die Kombination proxmox/lxc überschreibt beim Start diese Konfigurations-Dateien. Insbesondere wird der Rechnername auf die Adresse „127.0.1.1“ gesetzt. Dazu habe ich folgenden Workaround:

/Pool1/subvol-250-disk-1/etc/init.d/0hostresolv

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          hostresolv
# Required-Start:
# Required-Stop:
# Should-Start:      glibc
# Default-Start:     S
# Default-Stop:
# Short-Description: Set hosts and resolv.conf
# Description:       Set hosts and resolv.conf
### END INIT INFO

case "$1" in
  start|"")
    [ -e /etc/hosts ] && cp -a /etc/hosts /etc/hosts.SET
```

```
    [ -e /etc/hosts.OK ] && cp -a /etc/hosts.OK /etc/hosts
    [ -e /etc/resolv.conf ] && cp -a /etc/resolv.conf /etc/resolv.conf.SET
    [ -e /etc/resolv.conf.OK ] && cp -a /etc/resolv.conf.OK /etc/resolv.conf
    exit 0
    ;;
  stop)
    exit 0
    ;;
esac
exit 3
```

Aktivieren mit

```
chmod +x /Pool1/subvol-250-disk-1/etc/init.d/0hostresolv
```

/Pool1/subvol-250-disk-1/etc/resolv.conf.OK

```
domain intern.comasys.ch
nameserver 8.8.8.8
```

/Pool1/subvol-250-disk-1/etc/hosts.OK

```
127.0.0.1   localhost
62.2.169.250    template.local.comasys.ch template
::1    localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

CT Starten

Weiter in einem SSH-Terminal mit „ssh root@62.2.169.250 -p55“.

hosts / resolv update-rc.d 0hostresolv defaults Jetzt den CT gleich nochmals neustarten, damit unser neues Script auch ausgeführt wird.

Mit „ls -la /etc/{hosts,resolv.}*“ kontrollieren, ob die .SET erstellt wurden und die .OK kopiert wurden.

Update

```
apt-get update
apt-get upgrade
```

Notwendige Pakete

```
apt-get install
 attr
 acl
```

System-Konfiguration

```
dpkg-reconfigure dash
dpkg-reconfigure locales
dpkg-reconfigure tzdata
update-alternatives --config editor
```

Mail-Versand

In den nächsten Schritten gehe ich davon aus, dass wir über SSH auf dem CT arbeiten. Wer als Editor Mouspad verwenden will, muss die Anpassungen auf dem Host vornehmen. Der CT für das Template ist vom Host aus im Verzeichnis „/Pool1/subvol-250-disk-1/“ zu finden.

Damit die Cron-Jobs ihre Mail sauber abliefern können, müssen sowohl auf dem Host als auch in jedem einzelnen Container (auch in allen zukünftigen) noch Anpassungen vorgenommen werden.

Ich verwende in fast allen CT die gleiche Adresse. Deshalb muss ich in den neuen CT nichts mehr anpassen, wenn diese bereits im Template eingetragen sind.

Sender /etc/postfix/main.cf Folgende Zeile muss am Ende angefügt werden:

sender_canonical_maps=hash:/etc/postfix/sender_canonical /etc/postfix/sender_canonical Diese Datei wird neu erstellt.

root absender@domain www-data absender@domain Damit in Zukunft die Absender umgeschrieben werden (beim Empfänger wird als Absender 'absender@domain' erscheinen), muss das Map-File erstellt und postfix neu gestartet werden

postmap /etc/postfix/sender_canonical service postfix restart Ab sofort werden mails von root (also z.Bsp Cron-Jobs) mit dem neuen Absender verschickt. Das zweite Beispiel www-data wäre für den apache Web-Server, der hier aber nicht installiert ist.

Empfänger /etc/crontab Folgende Zeile muss am Anfang eingefügt werden:

MAILTO=„email@addresse“ /etc/aliases Damit eMails die z.bsp an 'root' oder 'postmaster' auf diesem Rechner gerichtet sind auf eine beliebige Adresse weitergeleitet werden, muss folgendes eingetragen werden:

root: weiter.an@diese.adresse postmaster: weiter.an@andere.adresse anschliessend das Map-File neu erstellen (und eventuell postfix neu starten)

postalias /etc/aliases service postfix restart Container Stoppen Das war es schon. Der CT muss jetzt gestoppt werden

halt Sobald die Verbindung getrennt ist, kann unser Template erstellt werden.

Template erstellen

(Ab jetzt wieder auf dem Host!)

Für das erstellen des Template habe ich mir ein Bash-Script gebastelt.

Ich lösche die nicht benötigen 'locale' und 'man'-Dateien. Nur englisch und deutsch bleibt, der Rest darf weg. Auch die ganzen doc- und info-Verzeichnisse können weg. Falls ich da mal was brauche ist es auf dem Host noch vorhanden, oder auch im Internet zu finden.

Nun wird der CT gestartet um ein Update/Upgrade durchzuführen, anschliessend die apt-Datenbanken zu säubern und die Datenbank für die man-Dateien neu aufzubauen.

Die Log-Files lösche ich auch, ebenso die /tmp und /var/tmp. Da ich noch nie verstanden habe, wieso es 2 temporäre gibt, lege ich für das /var/tmp einen Symlink auf /tmp an.

Die Netzwerk-Konfiguration wird beim erstellen eines CT neu angelegt, deshalb muss die 'interfaces' auch noch weg.

Jetzt muss nur noch das Template erstellt werden.

/Pool1/Save.sh

```
#! /bin/bash

ID=250

SRC=/Pool1/subvol-${ID}-disk-1
DST=/var/lib/vz/template/cache/Template.tar.gz

if [ ! -d ${SRC} ]
then
 echo "error!"
 exit
fi

TT=$( tempfile )

cd ${SRC}/..
echo "start container"
lxc-start --name ${ID}
echo "apt"
lxc-attach --name ${ID} -- apt-get update &>/dev/null
lxc-attach --name ${ID} -- apt-get upgrade -y &>${ID}-upgrade.log
lxc-attach --name ${ID} -- apt-get clean &>/dev/null
echo "mandb"
lxc-attach --name ${ID} -- mandb &>/dev/null
echo "stop container"
lxc-stop --name ${ID}

cd ${SRC}
echo "remove"
DIR=usr/share/locale
if [ -d "${DIR}" ]
then
 ls -1 ${DIR} >${TT}
 while read LINE
 do
```

```
  keep=0
  [ ${LINE:0:2} == "de" ] && keep=1
  [ ${LINE:0:2} == "en" ] && keep=1
  [ ${keep} -eq 0 -a -d "${DIR}/${LINE}" ] && rm -rf "${DIR}/${LINE}"
 done < ${TT}
fi
DIR=usr/share/man
if [ -d "${DIR}" ]
then
 ls -1 ${DIR} >${TT}
 while read LINE
 do
  keep=0
  [ ${LINE:0:2} == "de" ] && keep=1
  [ ${LINE:0:2} == "en" ] && keep=1
  [ ${LINE:0:3} == "man" ] && keep=1
  [ ${keep} -eq 0 -a -d "${DIR}/${LINE}" ] && rm -rf "${DIR}/${LINE}"
 done < ${TT}
fi
rm -f ${TT}

rm -rf usr/share/doc/*
rm -rf usr/share/doc-base/*
rm -rf usr/share/info/*

echo "cleanup"
rm -rf var/log/*
rm -rf tmp
rm -rf var/tmp
mkdir tmp
chmod 1777 tmp
ln -sf ../tmp var/tmp

rm -f etc/network/interfaces

echo "create archive"
[ -e "${DST}" ] && rm -f "${DST}"
tar -czf "${DST}" ./ 2>/dev/null
echo "...done"
```

Sollte ausführbar sein:

```
chmod +x /Pool1/Save
```

Wenn wir den nächsten CT erstellen, steht nebst dem Orginal auch unser neues Template.tar.gz zur Verfügung. Im weiteren Verlauf werde ich ausschliesslich das eigene verwenden, dann muss ich nich jedes mal meine Pakete enspielen und die Konfig (dash, locale, tzdata, editor) sind auch bereits erledigt.

Das obige Batch-Script kann auch periodisch (z.Bsp mit cron) ausgeführt werden und das Template

immer auf dem aktuellen Stand zu halten. Auf meinem Test-Server brauche ich immer mal wieder einen neuen CT, deshalb mache ich den Cron-Update jeweils um 1:00 am Samstag:

/etc/cron.d/templatesave

```
0 1 6 * * /Pool1/Save
```

Dieser CT wird natürlich nicht automatisch gestartet! Und wir lassen ihn auch ausgeschaltet.

Weitere Alternative

Software (nach)installieren

Ich gehe hier davon aus, dass folgende Pakete bereits in einem der vorherigen Schritte Einzug auf dem Server gefunden haben:

- mdadm
- lvm2
- xfsprogs

```
# aptitude install lxc
```

logisches Volume erstellen

Ich gehe hier davon aus, dass folgende Voraussetzungen bereits in einem der vorherigen Schritte konfiguriert wurden:

- eine Netzwerkbrücke (z. B. „br0“)
- ein Software-RAID1 (z. B. „/dev/md3“) als pysikalisches Volume für LVM2
- eine Volume-Gruppe für logische Volumen (z. B. „sys“)

```
# lvcreate -L2G -n lxc_template sys
```

Dateisystem erstellen und einhängen

```
# mkfs.xfs -L template /dev/sys/lxc_template
# mkdir /var/lib/lxc/template
# mount /dev/mapper/sys-lxc_template /var/lib/lxc/template
```

LinuX Container erstellen

```
# lxc-create -n template -t debian -- -r jessie
```

In der letzten Ausgabe des Befehls, erfährt man das root-Passwort für den Container.

Wer mag, kann es sich aufschreiben, in wenigen Minuten werden wir es jedoch selber „von außen“ ändern.

LXC anpassen

Netzwerkkonfiguration

```
# cat <<EOF >> /var/lib/lxc/template/config
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.ipv4 = 192.168.0.254/24
lxc.network.ipv4.gateway = 192.168.0.1
lxc.network.hwaddr = 02:34:56:78:90:fe
EOF
```

Debian Repository

```
# cat <<EOF > /var/lib/lxc/template/rootfs/etc/apt/sources.list
deb http://ftp.de.debian.org/debian jessie main contrib non-free
deb http://ftp.de.debian.org/debian jessie-updates main contrib non-free
deb http://security.debian.org/ jessie/updates main contrib non-free
EOF
```

(mein persönliches) "sauberes" Update-/Upgrade-Skript

```
# wget
https://raw.githubusercontent.com/casualscripter/debian-stuff/master/usr/loc
al/sbin/clean_upgrade \
  -O /var/lib/lxc/template/rootfs/usr/local/sbin/clean_upgrade
# chown root:root /var/lib/lxc/template/rootfs/usr/local/sbin/clean_upgrade
# chmod 700 /var/lib/lxc/template/rootfs/usr/local/sbin/clean_upgrade
```

/etc/hosts

```
# echo "192.168.0.254 template.domain template" >
/var/lib/lxc/template/rootfs/etc/hosts.tmp
# cat /var/lib/lxc/template/rootfs/etc/hosts" >>
/var/lib/lxc/template/rootfs/etc/hosts.tmp
# mv /var/lib/lxc/template/rootfs/etc/hosts{.tmp,}
```

/root/.bashrc

```
# mv /var/lib/lxc/template/rootfs/root/.bashrc{,_DEBIAN}
# cp /var/lib/lxc/template/rootfs/etc/skel/.bashrc
/var/lib/lxc/template/rootfs/root/.bashrc
```

```
# sed 's/HISTSIZE=1000/&0/' /var/lib/lxc/template/rootfs/root/.bashrc >
/var/lib/lxc/template/rootfs/root/.bashrc.tmp
# mv /var/lib/lxc/template/rootfs/root/.bashrc{.tmp,}
```

```
# sed 's/HISTFILESIZE=2000/&0/' /var/lib/lxc/template/rootfs/root/.bashrc >
/var/lib/lxc/template/rootfs/root/.bashrc.tmp
# mv /var/lib/lxc/template/rootfs/root/.bashrc{.tmp,}
```

```
# sed 's/#force_color_prompt=yes/force_color_prompt=yes/' >
/var/lib/lxc/template/rootfs/root/.bashrc.tmp
# mv /var/lib/lxc/template/rootfs/root/.bashrc{.tmp,}
```

Die Zeile

```
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\0
33[01;34m\]\w\[\033[00m\]\$ '
```

durch

```
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;31m\]\u@\[\033[01;35m\](lxc
)\[\033[01;31m\]\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
```

ersetzten (z. B. mit „nano“).

```
# sed "s/#alias grep='grep --color=auto'/alias grep='grep --color=auto'/" >
/var/lib/lxc/template/rootfs/root/.bashrc.tmp
# mv /var/lib/lxc/template/rootfs/root/.bashrc{.tmp,}
```

```
# sed "s/#alias ll='ls -l'/alias ll='ls -lisa'/" >
/var/lib/lxc/template/rootfs/root/.bashrc.tmp
# mv /var/lib/lxc/template/rootfs/root/.bashrc{.tmp,}
```

```
# echo 'HISTTIMEFORMAT="%F %T "' >>
/var/lib/lxc/template/rootfs/root/.bashrc
```

root-Passwort ändern

```
# echo "Change password for root in the LXC template:"
# passwd -R /var/lib/lxc/template/rootfs
```

öffentlichen SSH-Schlüssel (ggf. erstellen und) integrieren

Wenn man für root auf dem Host noch keinen SSH-Schlüsselpaar erzeugt hat, muss man es wie folgt

nachholen:

```
# ssh-keygen -b 2048 -t rsa -N "" -f /root/.ssh/id_rsa
```

Danach kopieren wir den öffentlichen Schlüssel in den LXC, damit wir uns später per SSH vom Host aus im Gast anmelden können:

```
# cp /root/.ssh/id_rsa
/var/lib/lxc/template/rootfs/root/.ssh/authorized_keys
```

LX Container convert privileged to unprivileged Container

```
vzdump 204 --exclude-path /var/spool/postfix/dev/random --exclude-path
/var/spool/postfix/dev/urandom --dumpdir /mnt/pve/backupvm/dump/
pct restore 204 vzdump-lxc-204-2019_05_04-13_59_11.tar --storage pool2_ct --
unprivileged
```