LTSP mit Ubuntu 12.04.3 LTS optimiert für Windows Terminalserver

Gitlab Project

GitlabProject can't find your project! Please check your **namespace** and your **project name**. Make sure you set up the right server and the right token!

Das Linux Terminal Server Project dient als eine Art Framework, um eine Umgebung einzurichten, die es erlaubt, einfache Thin Clients (Rechner mit minimaler Ausstattung, meist ohne Festplatte) über das Netzwerk zu starten und Anwendungen auf dem Server zu benutzen. Die Vorteile bestehen darin, dass die Administratoren nur den Hauptserver verwalten müssen. Sämtliche Anwendungen müssen nur einmal installiert werden, um jedem zur Verfügung zu stehen. Wenn ein Client ausfällt, kann einfach ein neuer aufgestellt werden, und der Benutzer kann in wenigen Minuten die Arbeit wieder aufnehmen.

Voraussetzungen für Mindestlauffähigkeit

Da sämtliche Aufgaben vom Client auf den Server übertragen werden, muss der Server über eine entsprechende Ausstattung verfügen. Dabei ist das Augenmerk weniger auf eine starke CPU zu legen als vielmehr auf Arbeitsspeicher und schnelle Festplatten. Je nach Anzahl der Clients sollte der Server über mehr als 1 GB RAM verfügen. Als Faustregel gilt 256 MB RAM für die Serveranwendungen und zusätzlich ~30 MB RAM für jeden angeschlossenen Client. Werden viele grafische Anwendungen gleichzeitig verwendet, sollte man pro Client mit ~128 MB RAM rechnen. Ein Server, der 5 Clients bedient, sollte also mindestens ~512 MB RAM besitzen, ein Server für 20 Clients entsprechend ~1024 MB. Auch auf eine gute Datenrate der Festplatte(n) ist zu achten. IDE-Platten versagen ihren Dienst ab etwa 10 Clients aufgrund des zu hohen Datenaufkommens. SCSI- oder SATA-Festplatten sollten aber mit 20 Clients fertig werden. Bei den Festplatten ist weiter darauf zu achten, dass bei vielen

Clients die Platten im Dauerlauf sind und somit mechanisch sehr beansprucht werden. Man sollte daher in Erwägung ziehen spezielle „Server“-Festplatten zu kaufen, die eine höhere Lebensdauer auch bei hoher Beanspruchung haben. Es ist auch zu empfehlen ein Netzwerk mit einer Datenübertragungsrate von 100 Mbit/s und mehr zu verwenden, da es aufgrund der X-Server Daten zu einem entsprechend hohen Datenverkehr kommt, und langsamere Netzwerke nicht mehr ausreichen.

Weiter sollte man beachten, dass die Clients, die über das Netzwerk booten wollen, einen Netzwerk-bootfähige Netzwerkkarte haben. Am besten wäre eine Karte, die entweder PXE oder Intel Etherboot unterstützt.

Installation

Eingesetztes Betriebssystem: Edubuntu 12.04 64bit und einem Client Image für 32bit. Natürlich fragen sich jetzt mache wieso 32bit. Hierfür gibt es mehrere gute Gründe.

- Noch teilweise Inkompatiblität zu Kauf- und OpenSource-Programmen (z.B. Dudenkorrektor)
- Hat man bei der Clientstruktur eine Mischstruktur, (64 und 32bit) so muss man 2 Images zur Auslieferung bereitstellen, und demnach auch 2 pflegen.
- Nachdem Prozessorgeschwindigkeit nicht das Riesenthema am Desktop ist, und die meisten Progamme die Kerne ja nicht ganz ausnutzen können, ist es nicht nötig ein 64bit System hierfür zu installieren. Und Ausnahmen wie „ffmpeg“ oder „Videoschnitt mit Cinelerre“ benötigen ohnehin einen Fullstateclient.

Nach erfolgreicher Installation eines Edubuntu müssen als ersters folgende Schritte ausgeführt werden:

- DHCP-Server falls am LTSP-Server nicht benötigt deinstallieren
- Das Clientimage nach seinen Bedürfnissen anpassen (nsswitch.conf, Avahi installiere, Domäne anpassen, Epoptes, rdesktop, usw)
- Rootpasswort setzen
- Wake on Lan einstellen (/etc/init.d/halt) **hier netdown auf „now“ setzten**.
- Um das Clientimage einfach einzubinden folgendes Script in **„/usr/local/bin“** anlegen. Wir benennen dies **„ltsp-enter-chroot“**.

```
#!/bin/bash
BASE=/opt/ltsp;
FILES=(`ls -I images $BASE/`);
ARCH="";

while :
do
        echo -e "\nThe following chroots were found:\n----------------------
--------------\n";
        X=0;
        for f in `ls -I images $BASE`
        do
                X=$(($X+1));
                echo -e "$X)\t$f"
```

```
        done

        echo -en "\nPlease select a chroot [1-$X]: ";
        read ARCH_NUM;

        if [[ $ARCH_NUM =~ ^[0-9]+$ && $ARCH_NUM -gt 0 && $ARCH_NUM -le $X
]]; then
                ARCH=${FILES[$(($ARCH_NUM-1))]}
                echo -e "Selected: $ARCH\n ";
                break;
        else
                echo -e "Invalid selection.\n";
        fi
done

ROOT="$BASE/$ARCH"
chroot "$ROOT" mount -t proc proc /proc || die "Not a valid chroot: $ROOT"
mount --bind /var/cache/apt/archives "$ROOT/var/cache/apt/archives"
cp /etc/resolv.conf "$ROOT/etc/"
export LTSP_HANDLE_DAEMONS=false
echo "Entering chroot $ROOT, type 'exit' to exit."
chroot "$ROOT" || true
unset LTSP_HANDLE_DAEMONS
umount "$ROOT/var/cache/apt/archives"

if ! umount "$ROOT/proc"; then
        echo "$ROOT/proc is in use, forcing unmount!"
        umount -l "$ROOT/proc"
fi
```

Clientimage anpassen

Als ersters wechseln wir in die Chrootumgebung für unsere Thinclients.

```
root@vsrv-ltsp:/usr/local/bin# ltsp-enter-chroot

The following chroots were found:
-----------------------------------

1)    i386

Please select a chroot [1-1]: 1
Selected: i386

Entering chroot /opt/ltsp/i386, type 'exit' to exit.
root@vsrv-ltsp:/#
```

Die Chroot können wir fast wie eine eigene Maschine behandeln. Also installieren wir relevante Dinge nach:

```
apt-get install avahi-daemon avahi-utils rdesktop linux-headers-3.13.0-30-
generic epoptes-client mesa-utils nload
htop openssh-server cups lm-sensors language-pack-gnome-de libnss-mdns acpi
acpi-support ethtool nano ltspfs
```

Nachdem wir die Kernelheaders auch installiert haben können wir den aktuellen Nvidiatreiber auch nachinstallieren.

```
apt-get install nvidia-current
```

Der Treiber mit mit Kmod im Kernel als Modul einkompiliert. Wichtig ist das wir in die **„/etc/rc.local“** also den Linuxautostart folgende Zeilen vor dem **„exit 0“** eintragen:

```
dhclient
avahi-daemon -D
```

Der DHCP-Client startet auch ohne diesen Eintrag, immerhin bootet die Maschine ja ohnehin per PXE, der Clienteintrag publiziert lediglich den Hostnamen dem DHCP-Server und der Wiederum den DNSserver. Der Avahidaemon (Zerconf) startet nur mit diesem manuellen Eintrag. Weiters noch Broadcast von der Systemdoku kopieren:

```
cp /usr/share/doc/avahi-daemon/examples/s* /etc/avahi/services/.
```

Nun ist es an der Zeit das Clientimage das erste mal zu aktualisieren.

```
ltsp-update-image -a i386
ltsp-update-sshkeys (Nur beim Ersten mal oder einem Subnetchange)
ltsp-finish-client
```

Der letzte Befehl enthält wieder ein kleines Script unter /usr/local/bin welches folgende Zeilen enthält:

```
#!/bin/bash
service openbsd-inetd restart
service tftpd-hpa restart
```

Clients würden jetzt (sofern unser Server als Bootserver einem DHCP eingetragen ist) schon booten. Bei dem ersten Bootvorgang werden wir aber noch bemerken das die Sprache falsch ist, Tastatur auf Englisch und einen proprietären Nvidiatreiber keine Spur ist. Unser Stichwort ist die LTS.conf.

LTS.CONF

Nachdem man den Server gemäß LTSP konfiguriert hat, kann man nun die Clienten davon booten lassen. Ab und zu sind jedoch noch einige Feineinstellungen nötig, damit man komfortabel mit den Clienten arbeiten kann. Dazu gehören zum Beispiel das verwendete Sprachschema, die Bildschirmauflösung und das Anbinden von Peripheriegeräten an die Clienten. Diese Optionen übergibt man mittels der Datei lts.conf, welche auf dem Server liegt. Falls man die Datei benötigt, so muss man sie mit einem Editor anlegen und bearbeiten.

```
nano /var/lib/tftpboot/ltsp/i386/lts.conf
```

Diese Datei könnte so aussehen:

```
# lts.conf, provided by Edubuntu installer
# see http://edubuntu.org/documentation for more information

[default]
  LDM_THEME=ubuntu
  LDM_DIRECTX=True
  X_NUMLOCK=True
  LANG=de_DE.UTF-8
  LANGUAGE=de_AT.UTF-8
  LDM_LANGUAGE=de_AT.UTF-8
  XKBLAYOUT=de
  X_COLOR_DEPTH=24

  XSERVER = nvidia
  MODULE_01 = nvidia
  X4_MODULE_01 = glx

  PRINTER_0_TYPE     = U
  PRINTER_0_DEVICE   = /dev/usb/lp0

[00:0c:76:81:c5:df]
  HOSTNAME=ltsp146
  LDM_AUTOLOGIN = True

#  SCREEN_8=ldm
#  LDM_AUTOLOGIN=True
#  LDM_USERNAME=
#  LDM_PASSWORD=
#  SCREEN_07="rdesktop 10.70.99.4"
#  RDP_OPTIONS = "-x l -f"
#  RDP_OPTIONS="-x l -f -d iteas --plugin rdpsnd -k de -n $HOSTNAME --data
disk:Speicher=/media/ -- "

#  LDM_GUESTLOGIN=True
#  LDM_LOGIN_TIMEOUT=10
#  LOCAL_STORAGE=True
#  SCREEN_07="rdesktop -d iteas -kde -x l -f -r sound:local -r
disk:Drives=/media/root -n $HOSTNAME 10.70.99.4"
```

Die Default beschreibt die Optionen die für alle Geräte gültig sind, der Teil mit der MACadresse beschreibt Optionen die nur für diesen Client gültig sind. Den Hostnamen kann man sich aussuchen. Gibt man keinen an, wird **„ltsp“ + die letzte Section der IP-Adresse** verwendet. Nach jeder Änderung der LTS.conf muss man den Befehle **„ltsp-finish-client“** ausführen, damit Änderung beim Nächsten Boot auch greifen.

Der auskommentierte Teil beschreibt Optionen für Terminals ohne einen Linuxdesktop, hier wird direkt in die RDPsession gebootet. Nachteil dieser Lösung ist aber das man sich um die Möglichkeit

der Fernwartung selbst kümmern muss. Epoptes ist hierfür nicht kompatibel.

Windows Terminalserver bequem einbinden

Nun beschäftigen wir uns weiter mit dem Thema wie man einen Windows Terminalserver die Umgebung schön einbringen kann. Folgende Dinge werden wir einrichten:

- Autologon der Clients mittels SSHkey oder mit Benutzer/Passwort
- Autostart von Terminalserververbindung mittels Rdesktop
- USB/Drucker/Sound Mapping für den Windowsserver
- Globabler Rollout von Unity Icons für alle Benutzer
- Einrichtung der Epoptes Fernwartung

Anlegen von Benutzer und Gruppen, Einrichtung SUDO-Rechte

Als Gruppen legen wir uns ltspuser an. Diese Gruppen bekommt dann alle nötigen Rechte. User mit „**ltsp**“ müssen natürlich in dieser Gruppe Mitglied sein. Bevor wir mit dem Anlegen von Usern fortfahren, sollte man sich überlegen was man denn dem Defaultprofil beifügen möchte. Diese Daten und Configs folgendem Verzeichnis beifügen:

```
/etc/skel
```

Also ersters gehen wir von einem Autologin mit einer DHCP-Range aus. Das ganze macht natürlich nur Sinn in einem Netz mit nur einem VLAN/Subnet. Denn hat man mehr als ein VLAN/Subnet, wird es zu Hostnamenüberscheidungen kommen. Wir gehen von einer DHCP-Range von 10.70.10.50 –> 200 aus. Dem zu Folge sieht unser Befehl so aus:

```
for a in $(seq 50 200) ; do u=ltsp$a ; sudo adduser --disabled-password --
gecos ,,, $u ; done
```

Und schon sind 150 Userkonten mit Benutzerprofil fix fertig erstellt. Nun werden SSHkeys für das Clientimage generiert

```
mkdir -p /opt/ltsp/i386/root/.ssh ; sudo chmod og-rwx
/opt/ltsp/i386/root/.ssh ; sudo ssh-keygen -t rsa -N '' -f
```

```
/opt/ltsp/i386/root/.ssh/id_rsa
```

Und nun das Gegenstück am Server:

```
mkdir -p /var/lib/ssh_authorized_keys
for a in $(seq 50 200) ; do u=ltsp$a ; sudo cp
/opt/ltsp/i386/root/.ssh/id_rsa.pub /var/lib/ssh_authorized_keys/$u ; sudo
chown root:$u /var/lib/ssh_authorized_keys/$u ; sudo chmod 0640
/var/lib/ssh_authorized_keys/$u ; done
```

Mit dem folgenden Befehl sieht man die Keys, hier auch die Rechte beachten.

```
ls -l /var/lib/ssh_authorized_keys
```

SUDO Rechte am Clientimage eintragen

An dieser Stelle tragen wir auch gleich Sudorechte für die Gruppe LTSPUSERS am Client ein, damit wir zum späteren Zeitpunkt herunterfahren, neustarten, und auch in der Lage sind einen USBstick an Windows durchzumappen. Wir betreten wieder die Chroot des Images und führen folgenden Befehl aus.

```
visudo
```

wir könnten natürlich auch einfach ein **„nano /opt/ltsp/i386/etc/sudoers“** eingeben. Hier hätten wir aber keinen Syntaxcheck und es bestünde die Möglichkeit sich aus dem System auszusperren.

```
%ltspusers ALL=NOPASSWD: /usr/bin/rdesktop
%ltspusers ALL=NOPASSWD: /sbin/shutdown
%ltspusers ALL=NOPASSWD: /sbin/reboot
```

Dies heist so viel wie das alle in der Gruppe „ltspusers“ diese drei Befehle ohne Passworteingabe ausführen dürfen.

SSH Server konfiugrieren

Nun öffnen wir die Konfigdatei des SSH-Servers tätigen folgenden Eintrag am Ende:

```
nano /etc/ssh/sshd_config
AuthorizedKeysFile /var/lib/ssh_authorized_keys/%u
```

Danach den SSH-Server neu starten und das Clientimage neu bauen. Und schon könne sich alle Client die im Netz per PXE mit diesem Server booten automatisch einloggen. Der richtig Eintrag hierfür in der LTS.CONF wäre:

```
LDM_GUESTLOGIN=True
LDM_LOGIN_TIMEOUT=5
```

Möglichkeit Nummer 2, User mit Passwort und Autologin, ohne SSH-Keys

Jetzt fragt man sich natürlich warum man das tun sollte. Das hängt sehr von der Beschaffenheit des Netzes ab. Hat man mehrer VLANs/Subnetze, wird es zu Hostnamenüberschneidung kommen. Als einfachste Lösung: Man kopiert den LTSP-Server in jedes VLAN und hat somit keine Profilüberscheidungen. Oder wir geben den Benutzern ein Passwort.

* Benutzer: HOSTNAME * PASSWORT: HOSTNAME

In der lTS.conf findet man dann folgende Einträge wieder:

```
[00:0c:76:81:c5:df]
  HOSTNAME=tc-bibliothek
  LDM_AUTOLOGIN = True
```

Ein Timeout können wir hier uns sparen da wir uns in Folge ein Shutdownbutton über DCONF auto-generieren.

Bashskripte und Desktopstarter für Rdesktop anlegen

Unsere Bashskripte für Rdesktop legen wir unter **„/usr/local/bin“** ab. Unsere Desktopstarter unter **„/usr/local/share/applications/„**. Sollte dieser Pfad noch nicht existieren muss er angelegt werden. Desktopstarter werden sofort in Unitydash angezeigt. Als erstes Beispiel die Verbindung zu einem Terminalserver im globalen Autostart.

```
nano /usr/local/bin/rdesktop_vsrv-term3_thinclient
```

```
#!/bin/bash
ltsp-localapps "sudo rdesktop -d hspts -N -kde -x m -f -r sound:local -r
disk:USB=/media/root 10.70.10.6"
```

```
chmod +x /usr/local/bin/rdesktop_vsrv-term3_thinclient
```

Die Optionen können natürlich je nach Eigenbedarf variieren. Der Desktopstarter hierfür würde so aussehen.

```
nano /usr/local/share/applications/rdesktop-vsrv-term3.desktop
```

```
[Desktop Entry]
Name=Rdesktop vsrv-term3
Type=Application
Comment=RDESKTOP Verbindung zum Terminalserver
Exec=sh -c "sleep 3; rdesktop_vsrv-term3_thinclient"
Icon=/usr/share/icons/oxygen/48x48/apps/krdc.png
Categories=Office;
```

Das Sleepcommand hat den Sinn das Login ein wenig zu verzögern, da der Desktop noch nicht geladen wäre wenn die Verbindung zum Windows Terminalserver steht. Das würde die Verbindung unterbrechen. Um das ganze noch in den globalen Autostart für X zu geben geht man wie folgt vor:

```
cd /etc/xdg/autostart
ln -s /usr/local/share/applications/rdesktop-vsrv-term3.desktop .
```

Bashskripte und Desktopstarter für Shutdown/Reboot anlegen

```
nano /usr/local/bin/ltsp_thinclient_shutdown
```

```
#!/bin/bash
ltsp-localapps "sudo shutdown -h now"
```

```
nano /usr/local/bin/ltsp_thinclient_reboot
```

```
#!/bin/bash
ltsp-localapps "sudo reboot"
```

Nicht vergessen sämtliche Skripte immer „ausführbar“ machen. Sonst funktionieren diese nicht!

```
nano /usr/local/share/applications/ltsp_thinclient_shutdown.desktop
```

```
[Desktop Entry]
Name=Computer Ausschalten
Type=Application
Comment=Schalten den Thinclient aus
Exec="ltsp_thinclient_shutdown"
Icon=/usr/share/icons/default.kde4/64x64/actions/system-shutdown.png
Categories=Office;
```

```
nano /usr/local/share/applications/ltsp_thinclient_reboot.desktop
```

```
[Desktop Entry]
Name=Computer Neustarten
Type=Application
Comment=Startet den Thinclient neu
Exec="ltsp_thinclient_reboot"
Icon=/usr/share/icons/default.kde4/64x64/actions/system-reboot.png
Categories=Office;
```

Bashskripte und Desktopstarter für Iconrollout

(Unityfavorites) anlegen

Der Sinn dahinter ist das man die Icons am Unitypanel gloablisieren kann. Somit kann man für alle User Icons und Starter vorgeben. Würde ein Benutzer Icons löschen oder hinzufügen, wird das beim Nächsten mal Einloggen wieder überschrieben. Das Stichwort hierfür heist **„gsettings“**. Unser Autostartskript sieht folgender Maßen aus:

```
#!/bin/bash
gsettings set com.canonical.Unity.Launcher favorites "[ 'clientinfo-
localapp.desktop', 'ltsp_thinclient_shutdown.desktop',
 'ltsp_thinclient_reboot.desktop', 'rdesktop-vsrv-term3.desktop', 'nautilus-
home.desktop', 'firefox.desktop',
 'libreoffice-writer.desktop', 'libreoffice-calc.desktop', 'ubuntu-software-
center.desktop' ]"
```

Je nachdem ob man eine Sektion hinzufügt oder entfernt, wird auch das Icon dahinter beim nächsten Login hinzugefügt oder entfernt. Das ganze kommt jetzt wieder in den Autostart für X hinein:

```
cd /etc/xdg/autostart
ln -s /usr/local/share/applications/set_unity_icons_global.desktop .
```

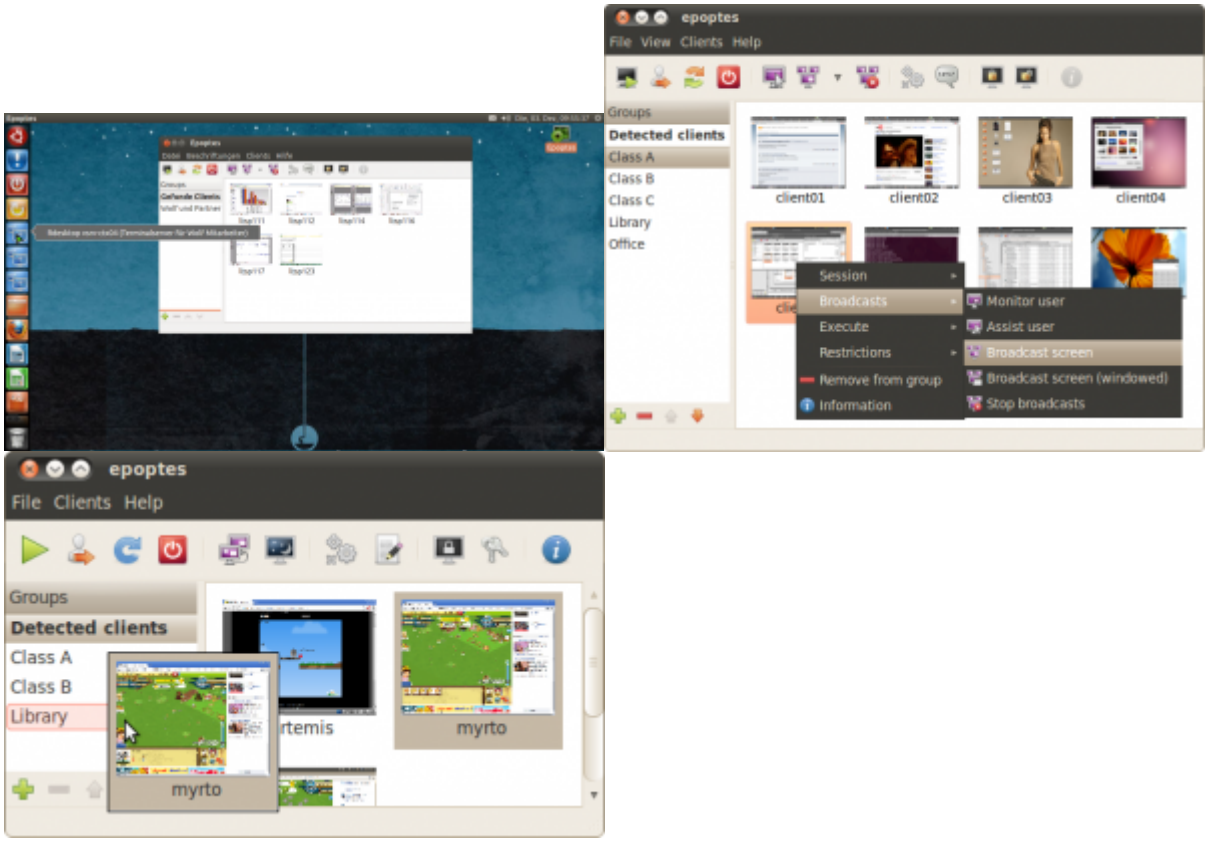
Nützliches brauchbares

Wenn man irgendwelche fragliche Änderungen am Clientimage durchführt, sollte man dieses vorher sichern:

```
cp -a /opt/ltsp /opt/ltsp_backup
cp -a /var/lib/tftpboot /var/lib/tftpboot_backup
```

- LTSP Handbuch zum Nachschlagen
- LDM Paket (Falls man Probleme mit Clientshutdown hat, dieses Paket am Clientimage einpielen)

Einrichtung der Epoptes Fernwartung



Die Fernwartung ist auf VNC basierend, fasst aber einige Möglichkeiten mehr zusammen.

- Sortierung der Clients
- Abholen von Maschineninfos
- Einschalten und Ausschalten der Clients (WOL)
- Befehle auf Client und auf Usershell ausführen
- Bildschirme anzeigen und für andere User freigeben
- Bildschrime sperren und übenehmen
- Nachrichten senden
- Sound kontrollieren
- Anwendbar auf Thin und Fatclients

Die Installation der Clientsoftware haben wir im oberen Abschnitt schon vollzogen. Nun werden Epoptes-Server installieren und konfigurieren.

```
apt-get install epoptes
```

Nach der Installation müssen wir die User die Zugriff auf die Fernwartung habne möchten zur Gruppe hinzufügen.

```
gpasswd -a iteasadm epoptes
```

Als nächstes nächstes installieren was Clientzertifikat am Clientimage:

```
chroot /opt/ltsp/i386
epoptes-client -c             # Fetches the OpenSSL certificate from the
server
exit
```

Nun das Image wieder neue bauen lassen:

```
ltsp-update-image -a i386
```

Auf Fatclients oder auch normale PCs installiert man auch das Zertifikat wie oben beschrieben. Zusätzlich muss man einen Hosteintrag setzen:

```
nano /etc/hosts
```

```
10.70.10.60      server
```

Die Installation wird hier auch noch genauer beschrieben.