OpenSUSE 42.3

ACHTUNG! Diese Anleitung wird nicht mehr gepflegt!

1) Paketauswahl mit Yast

Achtung! OpenSuse 42.3 enthält die Mono-Version 4.6. Bis einschließlich OpenSim 0.8.2 darf höchstens auf Mono 4.0.x aktualisiert werden. OpenSim 0.9.x, OpenSim (Metro) 0.8.3 und OpenSim Arriba sind auch unter Mono 4.6.x und damit unter OpenSuse 42.3 lauffähig, ältere OpenSim Versionen nicht!

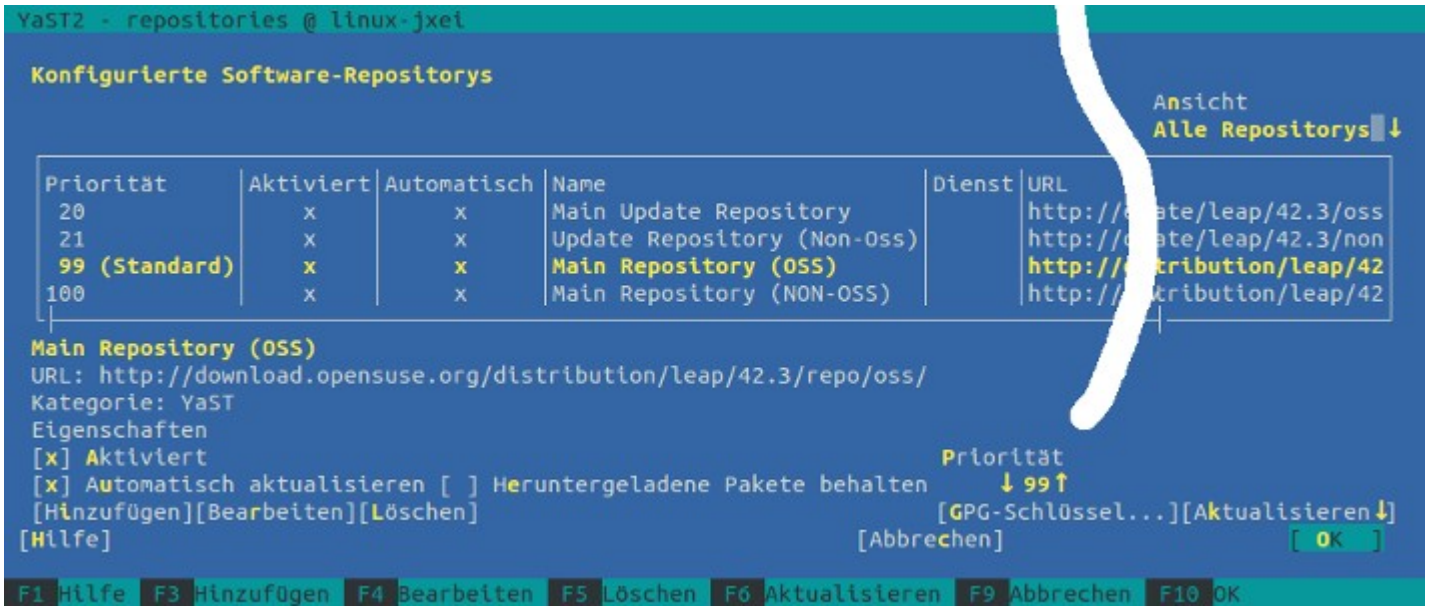
Falls der Provider eine "minimale" Default-Installation "OpenSuse 42.3" anbietet, kann diese als Basissystem übernommen werden, weiteres Abspecken lohnt den Aufwand nicht. Unabhängig vom Provider bietet OpenSuse über Yast in der Paketverwaltung ein Basissystem an unter "Filter -> Patterns", dann wähle "Base System" und zusätzlich "Yast System Administration".

Paket-Repositories in Yast einbinden

Menü "Software -> Software Repositories" auswählen, es öffnet sich der Dialog "Configured Software Repositories". Falls der Provider andere Einträge vorkonfiguriert hat, können die gelöscht werden. Insbesondere über FTP sind später bei geschlossenem FTP-Port eh keine Repositories mehr erreichbar. Danach richte die folgenden Repositories neu ein:

`http://download.opensuse.org/distribution/leap/42.3/repo/oss/` (Priorität 99)
`http://download.opensuse.org/distribution/leap/42.3/repo/non-oss/` (Priorität 100)
`http://download.opensuse.org/update/leap/42.3/oss/` (Priorität 20)
`http://download.opensuse.org/update/leap/42.3/non-oss/` (Priorität 21)

Eine Abbildung der englischen Version ist in der englischen Anleitung, so sieht der Dialog später nach der Sprachumschaltung aus:



Software-Pakete installieren

Empfehlenswert ist, zuerst ein Online-Update des Betriebssystems durchzuführen.

Menü "Software -> Software Management" auswählen, es öffnet sich der Dialog mit der Paketverwaltung. Jetzt die folgenden Pakete installieren:

=> `mariadb`
=> `mono-complete`
=> `tmux`

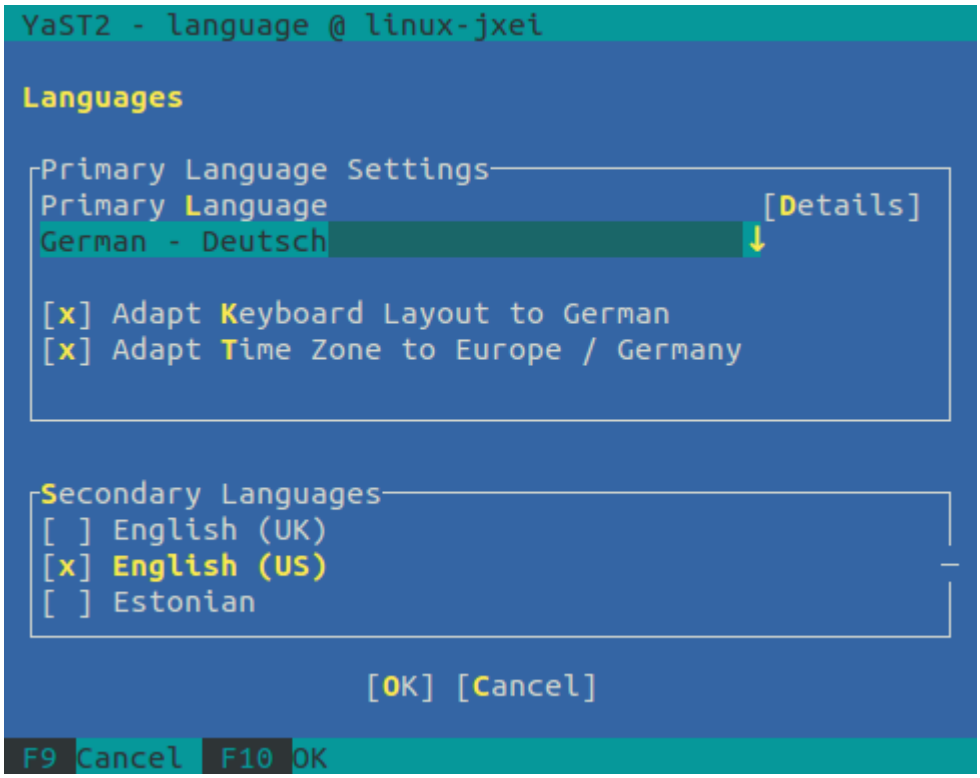
=> `nginx` (nur für Webserver gebraucht)
=> `php7-fpm` (nur für Webserver mit PHP-CGI Scripten gebraucht)
=> `php7-mysql` (nur für Webserver mit PHP und SQL Anbindung gebraucht, beispielsweise für CMS, Blogs oder Foren)

Durch die Auswahl werden viele weitere abhängige Pakete installiert.

2) Spracheinstellung in Yast

In Yast wähle "System -> Language".

=> Übers Menü: Primary Language "German", Kreuz bei Adapt Keyboard Layout to German, Kreuz bei Adapt Time Zone to Europe/Germany, Secondary Languages "English (US)"

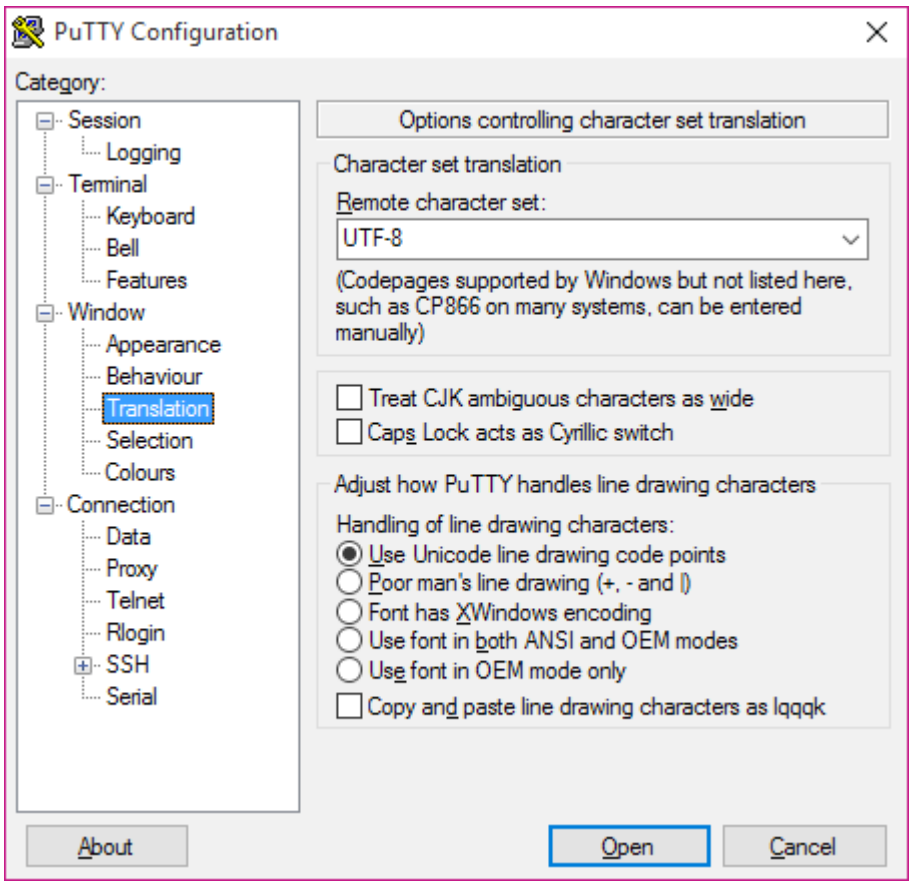


=> Untermenü "Details": Locale Settings for User root "Yes", Use UTF-8 Encoding ankreuzen und Detailed Locale Setting "de_DE".

=> Danach in the "/etc/sysconfig Editor" kontrollieren und von Hand nacheditieren. Im Untermenü System->Environment->Language:
RC_LANG=de_DE.UTF-8
RC_LC_ALL=""
RC_LC_...="" (alle leer)
ROOT_USES_LANG=yes
AUTO_DETECT_UTF8=no
INSTALLED_LANGUAGES=en_US,de_DE
Im Untermenü Hardware->Keyboard->KEYTABLE wähle "de-latin1.map.gz" (also ohne "nodeadkeys").

Spracheinstellung in PuTTY (Windows)

- Terminal/Keyboard: Function Keys and keypad: "Linux"
- Window/Translation: Character Set UTF-8
- Window/Translation: Handling of drawing characters: "Use Unicode line drawing..."
- Connection/Data: Terminal-type string: "linux"



Zeit einstellen in Yast

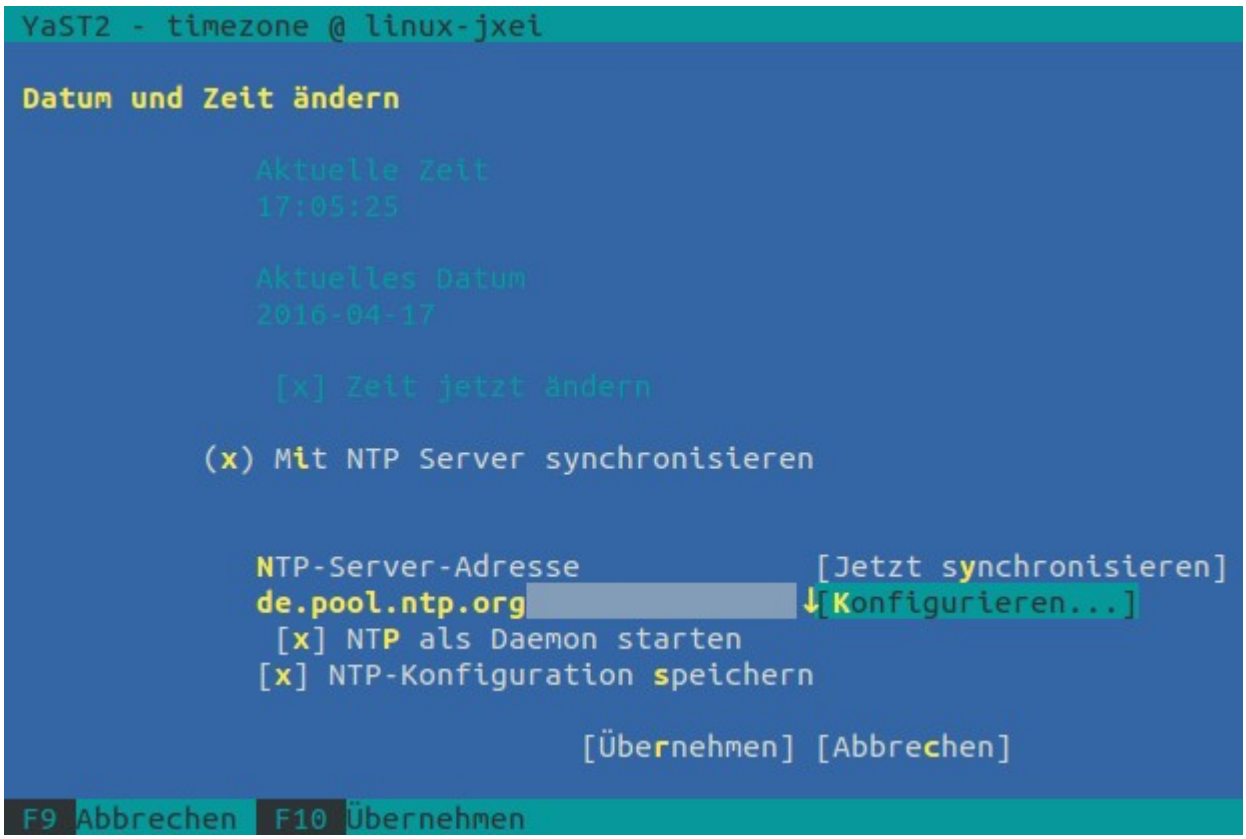
Zeitzone einstellen

Wähle das Menü "System -> Datum und Zeit", um den Dialog *Uhr und Zeitzone* zu öffnen. Du kannst durch Wählen einer Region und eines Landes jede beliebige Zeitzone wählen.

NTP Zeitsynchronisation

Hinweis: Wenn du einen virtuellen Server betreibst, ist jegliche NTP Synchronisation obsolet. Du darfst nicht die Hardwareuhr verstellen, und die Synchronisation wird (hoffentlich) vom Provider auf dem Hostsystem erledigt. Du kannst diesen Konfigurationsschritt überspringen, falls du auf einem virtuellen Server bist!

Rechts unten im Dialog "Uhr und Zeitzone" wähle [Andere Einstellungen...]. Im folgenden Dialog wähle:



=> Mit NTP-Server synchronisieren
=> NTP als Daemon starten
=> NTP-Konfiguration speichern

Falls die empfohlene NTP-Server-Adresse geändert werden soll, wähle eine Server-Pooladresse in der Nähe deines Landes. Mit [Konfigurieren ...] springen wir danach weiter in den Dialog *Erweiterte NTP-Konfiguration*, mit zwei Tabs. Im ersten Tab *Allgemeine Einstellungen* wähle (NTP-Daemon starten) "Jetzt und beim Systemstart". Im zweiten Tab *Sicherheitseinstellungen* wähle "NTP-Daemon iN Chroot-Jail ausführen".

Mit OK (um den erweiterten Dialog zu verlassen) and [Übernehmen] startet die soeben ausgewählte Konfiguration. Mit einem weiteren OK kann abschließend der Dialog *Uhr und Zeitzone* verlassen werden.

3) SSH härten

Die meisten scriptgesteuerten Angriffe zielen auf bekannte Ports und bekannte User. Deshalb stellen wir hier SSH auf einen unüblichen Port um, und beschränken den Zugang auf einen einzigen (namentlich genannten) User ohne root-Rechte.

Hinweis: Generell empfiehlt es sich, diese Konfiguration nur vorzunehmen, wenn ein aktuelles Backup des Gesamtsystems vorhanden ist. Sich mit SSH-Konfigurationsfehlern aus dem System auszusperren ist endgültig!
Notanker: Die meisten Serververmieter bieten (nach Kundenlogin) auf den Webseiten zur Serverkonfiguration die Möglichkeit, von einem Rettungssystem zu booten. Von dort sind Korrekturen an der Konfigurationsdatei noch möglich, das Rettungssystem mountet den gemieteten Server als gewöhnliche Festplatte.

Achtung! Den neuen Port vorher in der Firewall freigeben, sonst sperrst du dich selber aus!

1) In `/etc/ssh/sshd_config` den Port auf einen willkürlich gewählten umstellen, zum Beispiel Port 12345.
=> Nach erfolgreichem Verbindungstest über den neuen Port, den alten Port 22 in der Firewall schließen. *Hinweis:* Wird im Gegensatz zu dieser Anleitung doch die SuseFirewall verwendet, muss dort "ssh" aus der Liste der erlaubten Dienste gestrichen werden.

Achtung! Vor dem nächsten Schritt muss bereits im System ein weiterer Benutzer (ohne root Rechte) angelegt sein, sonst sperrst du dich selber aus! Also gegebenfalls mit Yast einen neuen (gewöhnlichen) Benutzer einrichten.

2) In `/etc/ssh/sshd_config` folgende 5 Parameter anpassen und davor die Kommentarzeichen entfernen:
=> Parameter "LoginGraceTime" auf "2m" setzen, also 2 Minuten Zeit für Loginversuche
=> Parameter "PermitRootLogin" auf "no" setzen
=> Parameter "MaxAuthTries" auf "3" Loginversuche setzen
=> Parameter "MaxSessions" auf maximal "1" gleichzeitig eingeloggten User setzen.
=> Parameter "AllowUsers" auf den speziell eingerichteten Benutzer (ohne root Rechte) setzen

Hinweis: Der Parameter "AllowUsers" ist in der Beispieldatei nicht vorhanden, also muss die Zeile `AllowUsers` von Hand eingefügt werden.

Tests:
=> Den Server einmal durchbooten und schauen, ob du mit dem neuen Account einloggen kannst. Wenn nicht, Backup einspielen...
=> Versuchen, als Benutzer root einzuloggen.

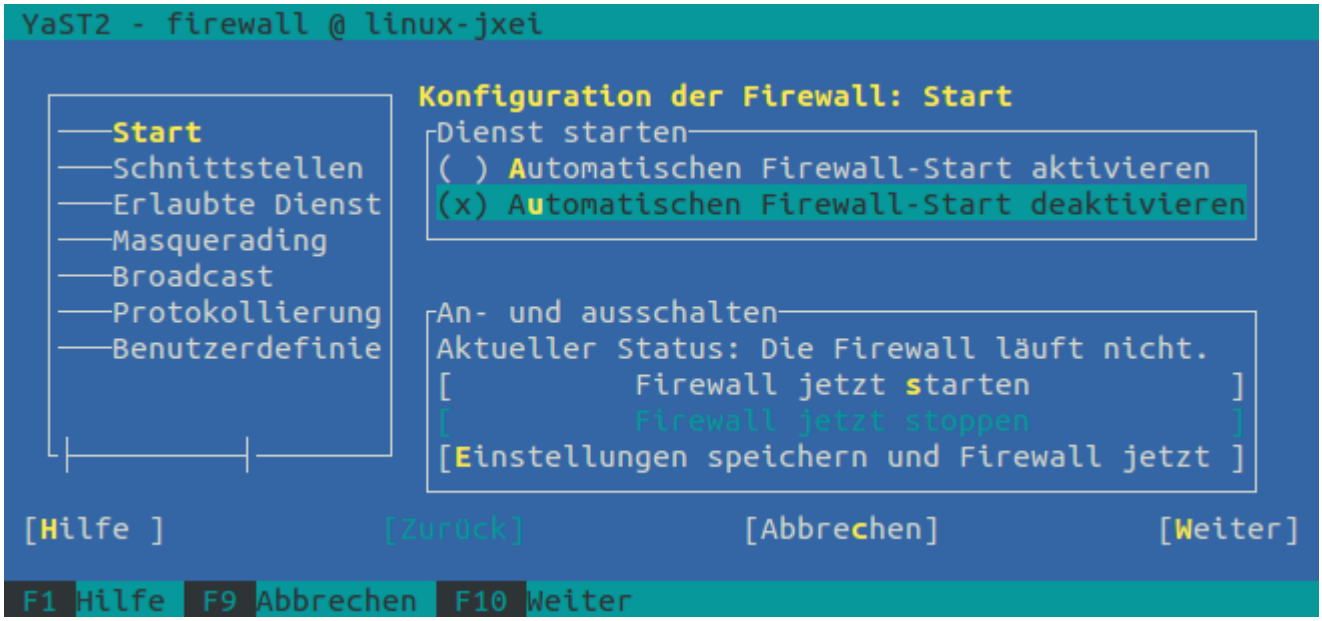
Hinweise: Es ist möglich, sich mit mehreren Konsolenfenstern einzuloggen. Alle Logins des selben Users vom selben Rechner aus gelten als eine Session. Wenn du erst mal remote eingeloggt bist, kann lokal auf dem Server der User mit dem Kommando `su NeuerUser` gewechselt werden. (Der neue User muss auf dem System bereits einen Account haben.

4) Firewall einrichten

Die über Yast konfigurierbare SuseFirewall2 erzeugt eine undurchsichtige Vielzahl von Filterregeln, aber ermöglicht nicht das Schließen von Ports für vom eigenen Rechner ausgehende Verbindungen. OpenSim muss im derzeitigen Entwicklungsstadium noch als unsicher eingestuft werden, denn das Suchen nach Sicherheits-Schwachstellen ist noch zweitrangig gegenüber der Implementierung und Stabilisierung der Funktionalitäten. Deshalb ist es sinnvoll, für eingeschränkte Accounts alle nicht zwingend benötigten Ports zu schließen. Dies erschwert einem Angreifer, den Server mit Hilfe nachinstallierter Software für Angriffe auf Drittrechner zu benutzen ... zumindest, so lange er keine Administrator-Rechte erlangt.

Aus obigen Gründen wird in dieser Anleitung die im Linux-Kernel eingebaute Firewall ohne Zuhilfenahme eines Tools konfiguriert. Dazu sind als Benutzer root drei Schritte erforderlich:

1) Die SuseFirewall in Yast abschalten, damit sie beim Rechnerstart nicht mehr aktiviert wird: "Sicherheit und Benutzer -> Firewall". Falls hier eine Fehlermeldung wegen fehlender Pakete erscheint, ist wahrscheinlich das Paket *SuSEfirewall2* noch nicht installiert. In diesem Fall erübrigt sich das Abschalten natürlich.



2) Folgendes Bash-Script mit Rechten ausschließlich für root abspeichern (chmod 700), zum Beispiel im Verzeichnis /root.

```
#!/bin/bash

# durch den Provider vorgegebener Name der Netzwerk-Schnittstelle
interface="venet0"

#### IPv6 Konfigurationsblock ####

# alle alten Regeln löschen
ip6tables -F

# alle Ports schließen
ip6tables -P INPUT DROP
ip6tables -P OUTPUT DROP
ip6tables -P FORWARD DROP
```

```
#### IPv4 Konfigurationsblock ####

# alle alten Regeln löschen
iptables -F

# alle Ports schließen
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# spezielle Steuersignale für IPv4 Kommunikation
iptables -A INPUT -p icmp --icmp-type destination-unreachable -j ACCEPT
iptables -A INPUT -p icmp --icmp-type time-exceeded -j ACCEPT
iptables -A INPUT -p icmp --icmp-type parameter-problem -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT

# lokalen Datenverkehr erlauben, notwendig für das Betriebssystem!
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# bereits bestehenden Verbindungen das Antworten erlauben
iptables -A INPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT

# eingehend SSH erlauben, eventuell Portnummer anpassen (für Fernwartung)
iptables -A INPUT -i ${interface} -p tcp --dport 22 -m conntrack --ctstate NEW -
j ACCEPT

# ausgehend DNS erlauben (für Auflösung von Domainnamen)
iptables -A OUTPUT -o ${interface} -p udp --dport 53 -m conntrack --ctstate NEW
-j ACCEPT
iptables -A OUTPUT -o ${interface} -p tcp --dport 53 -m conntrack --ctstate NEW
-j ACCEPT

# ein- und ausgehend HTTP erlauben (für SW-Updates und Webserver)
iptables -A INPUT -i ${interface} -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -o ${interface} -p tcp --dport 80 -j ACCEPT

# ausgehend NTP erlauben (für Zeitsynchronisation, entfällt bei virtuellen
Servern)
iptables -A OUTPUT -o ${interface} -p udp --dport 123 -m conntrack --ctstate NEW
-j ACCEPT

# Hier weitere Ports nach Bedarf öffnen analog zu SSH und DNS. Jeweils anpassen:
# => Richtung "INPUT -i" (einwärts) oder "OUTPUT -o" (auswärts)
# => Protokoll "udp" oder "tcp"
# => Portnummer

# zentrale OpenSim Dienste
iptables -A OUTPUT -o ${interface} -p tcp --dport 8002 -m conntrack --ctstate
NEW -j ACCEPT
iptables -A OUTPUT -o ${interface} -p tcp --dport 8003 -m conntrack --ctstate
NEW -j ACCEPT
# Die folgenden 5 Zeilen nur für das Metropolis Grid
iptables -A OUTPUT -o ${interface} -p tcp --dport 8000 -m conntrack --ctstate
NEW -j ACCEPT
iptables -A OUTPUT -o ${interface} -p tcp --dport 8001 -m conntrack --ctstate
NEW -j ACCEPT
iptables -A OUTPUT -o ${interface} -p tcp --dport 8004 -m conntrack --ctstate
NEW -j ACCEPT
iptables -A OUTPUT -o ${interface} -p tcp --dport 8005 -m conntrack --ctstate
NEW -j ACCEPT
iptables -A OUTPUT -o ${interface} -p tcp --dport 8006 -m conntrack --ctstate
NEW -j ACCEPT
```

```
# eigener Server
iptables -A INPUT -i ${interface} -p tcp --dport 9000 -m conntrack --ctstate NEW
-j ACCEPT
# eigene Regionen
iptables -A INPUT -i ${interface} -p udp --dport 9001 -m conntrack --ctstate NEW
-j ACCEPT
iptables -A INPUT -i ${interface} -p udp --dport 9002 -m conntrack --ctstate NEW
-j ACCEPT
# alle fremden akzeptierten Regionen, auch Hypergrid
iptables -A OUTPUT -o ${interface} -p tcp --dport 9000:65535 -m conntrack --
ctstate NEW -j ACCEPT
```

3) Einen Aufruf des Scriptes in den Autostart einbauen (dort bereits berücksichtigt).

Achtung! Wenn nach einem Rechnerstart weder SuseFirewall2 läuft noch dieses Script aufgerufen wurde, dann ist der Rechner völlig ungeschützt! Deshalb ist empfehlenswert, bis zur fertigen Konfiguration auch des Autostarts möglichst außer SSH keine vom Internet zugänglichen Dienste zu starten. Und nicht vergessen, die Firewall zu testen!

Hinweis: Generell empfiehlt es sich, diese Konfiguration nur vorzunehmen, wenn ein aktuelles Backup des Gesamtsystems vorhanden ist. Starte die Firewall manuell, bevor du sie in den Autostart einbaust! Es ist durchaus möglich, sich mit Firewall-Konfigurationsfehlern endgültig aus dem System auszusperren.
Notanker: Die meisten Serververmieter bieten (nach Kundenlogin) auf den Webseiten zur Serverkonfiguration die Möglichkeit, von einem Rettungssystem zu booten. Von dort sind Korrekturen an der Konfigurationsdatei noch möglich, das Rettungssystem mountet den gemieteten Server als gewöhnliche Festplatte.

5) Autostart einrichten

Annahme in dieser Anleitung: Der eingeschränkte Benutzer für OpenSim heißt "maria".

Wir kopieren das folgende Skript mit beispielsweise dem Dateinamen "maria-autostart.sh" als user *root* nach `/etc/systemd/system` . (Es kann ein beliebiges Verzeichnis gewählt werden, wir wählen hier der Übersichtlichkeit halber das Verzeichnis mit der Konfigurationsdatei.) Mit diesem Script wird beim Rechnerstart das selbstgemachte Firewall-Script aufgerufen. Schließlich wird der Benutzer auf den eingeschränkten OpenSim-Account gesetzt (hier maria), und in dessen Home-Verzeichnis eine Datei "autostart.sh" aufgerufen. Über die autostart.sh kann später der eingeschränkte Benutzer beim Hochlauf selber Programme starten.

```
vi /etc/systemd/system/maria-autostart.sh

#!/bin/sh
#
# /etc/systemd/system/maria-autostart.sh

case $1 in
  start)
    bash /root/firewall.sh
    su maria -l -c 'bash autostart.sh' &
    ;;
  stop)
    ;;
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac
```

Danach setzen wir die Ausführungsrechte:

```
chmod u+x /etc/systemd/system/maria-autostart.sh
```

Damit unser Script als Service gestartet werden kann, benötigt es noch eine Konfigurationsdatei. Hier ist das Verzeichnis `/etc/systemd/system` zwingend, die Dateiendung `.service` ebenso. Also nennen wir die Konfigurationsdatei sinnvoll "maria-autostart.service":

```
vi /etc/systemd/system/maria-autostart.service

[Unit]
Description=At system boot start firewall and call autostart.sh of user.

[Service]
Type=oneshot
ExecStart=/etc/systemd/system/maria-autostart.sh start
ExecStop=/etc/systemd/system/maria-autostart.sh stop
RemainAfterExit=yes

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

Wenn beide Dateien erstellt sind, muss der neue Service noch aktiviert werden:

```
systemctl enable maria-autostart.service
```

Nun logge mit dem eingeschränkten OpenSim-Account ein und erstelle in dessen Home-Verzeichnis (hier maria) eine vorerst leere Textdatei mit dem Namen "autostart.sh". Diese Datei wird später mit den Startbefehlen für OpenSim gefüllt. Mache dies nicht als User root, weil sonst später der eingeschränkte User die Datei nicht ändern kann! Setze Ausführungsrechte: "chmod u+x autostart.sh"

6) Rechner einmal täglich booten

OpenSim ist ja noch im Beta-Stadium, und läuft manchmal nicht ganz stabil. Deshalb soll der Rechner jeden Tag einmal neu starten. Füge dafür in der Datei /etc/crontab folgende Zeile ein:

```
0 6 * * * root shutdown -r now
```

Die Zahlen geben die Uhrzeit an, hier 6:00 Uhr.

7) MariaDB (oder MySql) konfigurieren

Zuerst erzwingen wir den Zeichensatz UTF8 zur internen Datenspeicherung, um mögliche Codepage-Probleme zu umgehen.

```
vi /etc/my.cnf
```

In den Sektionen [client], [mysql] und [mysqld] ergänzen wir die folgende Zeilen:

```
[client]
...
default-character-set=utf8
...
[mysql]
...
default-character-set=utf8
...
[mysqld]
...
collation-server=utf8_unicode_ci
init-connect='SET NAMES utf8'
character-set-server=utf8
...
```

Dann müssen wir den MySql-Server *MariaDB* aktivieren. Wir starten ihn daraufhin von Hand, weil der Dienst sonst erst beim nächsten Systemstart mit gestartet würde.

```
systemctl enable mysql.service
systemctl start mysql.service
```

Nun durchlaufen wir ein Konfigurationsscript, das einige unsichere Einstellungen entfernt.

```
/usr/bin/mysql_secure_installation
```

Das Script stellt einige Fragen, die Benutzereingaben sind fett dargestellt:

1) **(Enter)** - Das aktuelle root-Passwort eingeben, bei einer erstmaligen Installation einfach leer lassen (Enter drücken).

2) **Y** - Ja, Du willst sicherlich ein root-Passwort setzen.

3) **MyRootPassword** - Zweimal das neue root-Passwort eingeben, dieses irgendwo aufschreiben und an sicherem Ort lagern.

4) **Y** - Den anonymen User löschen: Ja! Dieser wurde nur für Tests angelegt, und wird im Produktivbetrieb nicht gebraucht. Mit dem anonymen User hätte jeder ohne Login Zugang zum Datenbankserver.

5) **Y** - Remote root-Logins verbieten! Auf die Datenbank wird *immer* lokal zugegriffen. Die Verbindung ins Internet wird nicht über die Datenbank hergestellt, sondern über OpenSim oder SSH.

6) **Y** - Testdatenbank entfernen.

7) **Y** - Privilegientabellen neu laden, damit die Änderungen sofort wirksam werden.

Tipps: Den aktuellen Status des MySql-Servers abfragen, ihn stoppen und neu starten geht mit den folgenden Befehlen.

- Start des Dienstes: "systemctl start mysql.service"
- Stopp des Dienstes: "systemctl stop mysql.service"
- Statusabfrage: "systemctl status mysql.service"

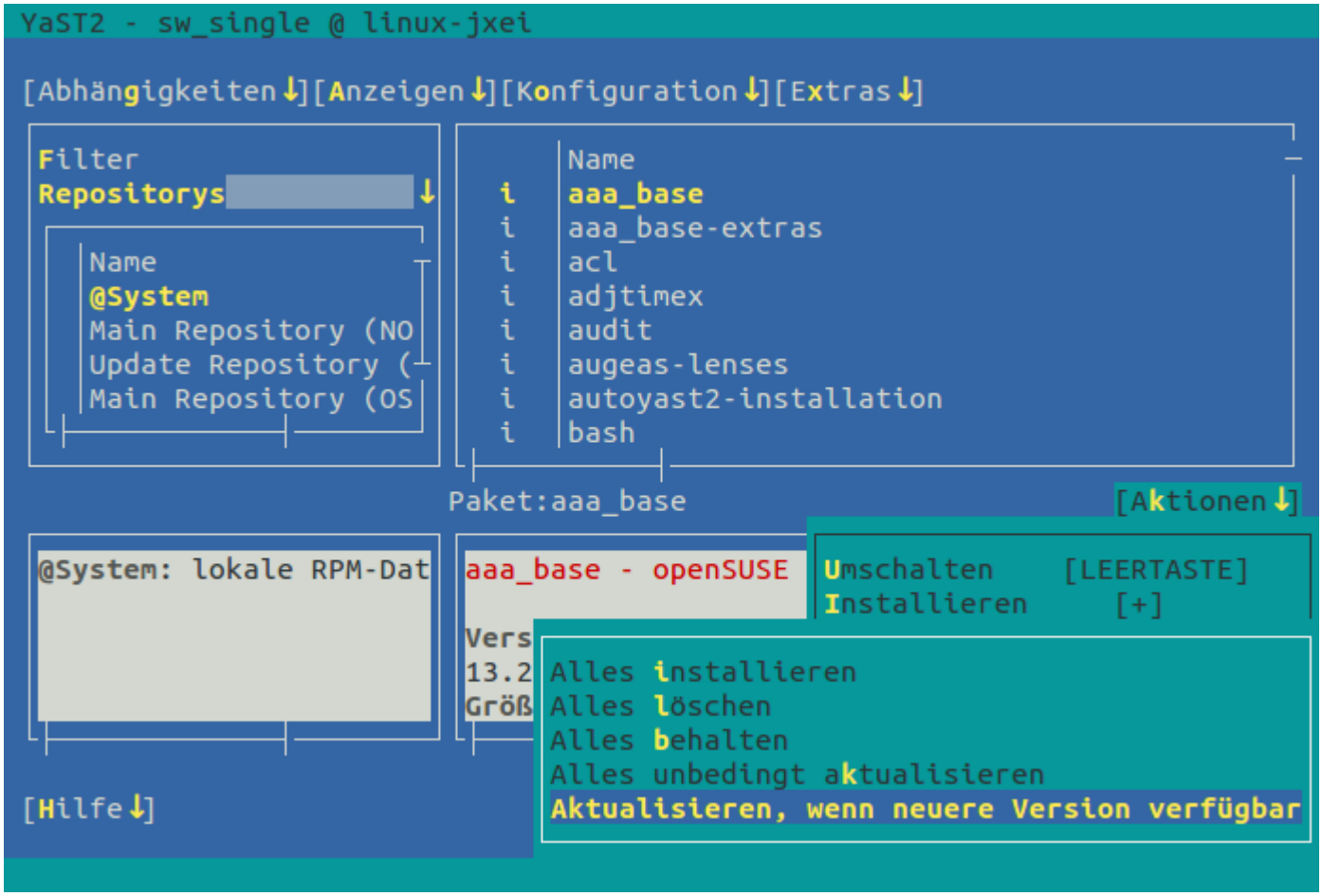
Nun wird eine neue Datenbank "opensim" und ein eingeschränkter Benutzer "opensim" angelegt. (Datenbank und Benutzer dürfen beliebig heißen, aber so werden sie später in der OpenSim-Anleitung genannt.) Die einzugebenden Kommandos sind im folgenden Screenshot fett dargestellt.

```
mysql -u root -p
Enter password: MyRootPassword
Welcome to the MariaDB monitor ...

MariaDB [(none)]> create database opensim;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> create user opensim identified by 'MyOpensimPassword';
Query OK, 0 row affected (0.00 sec)

MariaDB [(none)]> grant all privileges on opensim.* to opensim;
Query OK, 0 row affected (0.00 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 row affected (0.00 sec)

MariaDB [(none)]> quit;
Bye
```

8) Online-Updates

YaST bietet zwar auch die Möglichkeit, eine automatische Aktualisierung einzurichten. Aus Gründen der Betriebssicherheit (Verfügbarkeit) habe ich mich aber für rein manuelle Updates entschieden:

1. Yast aufrufen, dann "Software", "Software installieren oder löschen".
2. Im Dialog auswählen: "Filter", "Repositorys", "@System".
3. Weiter: "Aktionen", "Alle aufgelisteten Pakete...", "Aktualisieren, wenn neuere Version verfügbar".
4. "Übernehmen" (unten rechts), eventuelle abhängige Pakete akzeptieren.



9) Aktuelleres Mono installieren (optional)

(Zur Vertiefung siehe auch: Install Mono on Linux)

Um das aktuelle Mono aus dem Xamarin Repository zu installieren, muss zuerst ein neues Repository eingetragen werden (siehe Paketauswahl):

`http://download.mono-project.com/repo/centos/`

Damit Yast das Repository akzeptiert, deaktiviere temporär alle anderen Repositories. Im Gegenzug deaktiviere später das Mono-Repository, damit das Fremdpaket nicht bei normalen OpenSUSE-Updates berücksichtigt wird. Solange das Xamarin Repository eingebunden ist, kann die aktuellste Mono-Version installiert werden:

=> `mono-complete`

Durch die Auswahl werden viele weitere abhängige Pakete installiert, die in dem Repository enthalten sind. Einzelne Abhängigkeiten können innerhalb des Xamarin Repositories nicht aufgelöst werden. Dort wird zur Auswahl angeboten, die fehlenden Pakete aus dem (deaktivierten) OpenSUSE Repository zu installieren: Ja, die Pakete sollen von OpenSUSE genommen werden!