ssh

ssh in firewalld freischalten

Enable firewall rule for ssh.

```
sudo firewall-cmd --permanent --add-service=ssh
success
```

Reload firewall rules.

```
sudo firewall-cmd --reload
success
```

oder Firewall abschalten:

```
systemctl stop firewalld.service
```

Einsatzzwecke von ssh

- Fernadministration
- Dateien übers Netz kopieren
- Tunnel bauen

Grundfunktionen

Einloggen auf <Rechner> mit <Benutzer>: [1)]

```
ssh <Benutzer>@<Rechner>
```

hängende ssh-Verbindung beenden:

```
~.
```

ssh-Verbindung pausieren:

```
~Strg+z
```

Dateien auf andere Rechner kopieren: [2)]

```
scp <Datei> <Benutzer>@<Rechner>:/<Verzeichnis>
```

Dateien auf andere Rechner ohne lange Pfadangaben ins $HOME-Verzeichnis von <Benutzer>

kopieren:

```
scp <Datei> <Benutzer>@<Rechner>:
```

Syntax sshd prüfen

Vor dem Neustart des Dienstes die Konfigurationsdatei auf Syntaxfehler prüfen:

```
sshd -t
```

oder ausführlicher:

```
sshd -dddt
```

Server Keys

Fingerprint Hostkey überprüfen

```
ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
```

Server Key aus known_hosts entfernen

Name und IP-Adresse aus known_hosts entfernen:

```
ssh-keygen -R notebook02
ssh-keygen -R 192.168.1.202
```

Server Keys neu erzeugen

löschen:

```
rm /etc/ssh/ssh_host_*key*
```

Debian

```
dpkg-reconfigure openssh-server
```

SuSE / RedHat

```
service sshd restart
```

Public-Private-Key Authentifizierung

Als Nutzer Schlüsselpaar erzeugen

Einfach:

```
ssh-keygen
```

oder für Schlüssel mit Kommentar auf Basis von Curve25519

```
ssh-keygen -t ed25519 -C "Kommentar"
```

Public-Key auf anderen Rechner übertragen

```
ssh-copy-id -i .ssh/id_ed25519.pub nutzer05@notebook06
```

oder von Hand:

```
ssh nutzer05@notebook06 'mkdir -m 700 ~/.ssh'
ssh nutzer05@notebook06 'cat >> .ssh/authorized_keys' <
~/.ssh/id_ed25519.pub
ssh nutzer05@notebook06 'chmod 600 .ssh/authorized_keys'
```

Serverseitig Passwortauthentifizierung abschalten

/etc/ssh/sshd_config

```
UsePAM no
PasswordAuthentication no
```

Alternativ (CentOS 7)

/etc/ssh/sshd_config

```
UsePAM yes
ChallengeResponseAuthentication no
```

```
  PasswordAuthentication no
```

ssh-agent

Schlüssel dem `ssh-agent` hinzufügen:

```
ssh-add .ssh/id_dsa
```

[3)]

Prüfen, welche Schlüssel der agent kennt:

```
ssh-add -l
```

Nutzer Zugriff beschränken

mit authorized_keys command

Problem: die Datei ~/.ssh/authorized_keys gehört dem Nutzer, den man einschränken möchte.

~/.ssh/authorized_keys

```
restrict,command="/usr/bin/who" ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIHBqUtiLsRTLKquoVXKwhrPRD92CzaN9EOkVEfWoHfdC
nutzer26@notebook26
```

mit Match

/etc/ssh/sshd_config

```
…
Match User nutzer12
  ForceCommand /usr/bin/who
```

Fehlersuche

```
ssh -v nutzer06@notebook06
```

```
ssh -vv nutzer06@notebook06
```

SSH über mehrere Hops

```
+--------+                  +------------+                 +------------+
| ssh    |                  | Hop Server |                 | server     |
| client |  _ _ _ ssh _ _ _ | notebook01 |                 | notebook02 |
|        | ================ssh========tunnel============> |            |
+--------+  ‾ ‾ ‾ ‾ ‾ ‾ ‾ ‾ +------------+                 +------------+
```

Client

centos 7.4, debian 9, ubuntu 18.04: (ab OpenSSH-Version 7.3)

```
ssh -J notebook01.linuxhotel.de notebook02.linuxhotel.de
```

mit einer `~/.ssh/config` reduziert sich das zu:

~/.ssh/config

```
Host notebook02
  Hostname notebook02.linuxhotel.de
  ProxyJump notebook01.linuxhotel.de
```

```
ssh notebook02
```

Bei älteren SSH-Versionen:

centos 7, debian 8, ubuntu 16.04:

```
ssh -o ProxyCommand="ssh -W %h:%p notebook01.linuxhotel.de"
notebook02.linuxhotel.de
```

mit einer `~/.ssh/config` reduziert sich das zu:

~/.ssh/config

```
Host notebook02
  Hostname notebook02.linuxhotel.de
  ProxyCommand ssh -W %h:%p notebook01.linuxhotel.de
```

```
ssh notebook02
```

Bei noch älteren SSH-Versionen muss zusätzlich `netcat` bzw. `nc` auf dem Jumphost installiert sein:

debian 6, centos 6:

~/.ssh/config

```
Host notebook02
  Hostname notebook02.linuxhotel.de
  ProxyCommand ssh -q notebook01.linuxhotel.de nc %h %p
```

Hop Server

optionale Absicherung, damit sich der Benutzer nicht auf dem Hop-Server einloggen kann

/etc/ssh/sshd_config

```
# nutzer02 wird auf notebook02 weitergeleitet:
Match User nutzer02
  ForceCommand ssh nutzer02@notebook02 $SSH_ORIGINAL_COMMAND

# nutzer01 wird auf den mit der Option -W gewählten Server
weitergeleitet:
Match User nutzer01
  ForceCommand ssh %u@%h $SSH_ORIGINAL_COMMAND
```

~/.ssh/authorized_keys

```
restrict,port-forwarding ssh-rsa
AAAAB3NzaC1yc2EAXXXXXXXXXXXXXXXXXXXXXXXXX
```

Agent-Forwarding nutzen

```
ssh -A notebook17
ssh notebook07
```

besser, mit Konfigdatei:

~/.ssh/config

```
Host notebook17
  ForwardAgent yes
  IdentityFile ~/.ssh/id_rsa
```

```
  IdentitiesOnly yes
```

Sicherheitseinschränkung bei Agent-Forwarding

Sockets zu den ssh-agents anzeigen:

```
ls -ld /tmp/ssh-*/agent*
```

root kann den Socket für das Agent-Forwarding mitnutzen: [4)]

```
PIDS="$(lsof -t -a -i -c sshd -u nutzer05)"
```

[5)]

```
export SSH_AUTH_SOCK=$(for pid in $PIDS; do echo /tmp/ssh-*/agent.$pid; done
| head -n 1)
```

```
ssh nutzer05@notebook07
```

Auch ein ssh-agent auf dem client-Rechner kann ähnlich mitgenutzt werden:

```
PIDS="$(lsof -t -w -a -U -c /agent/ -u nutzer05)"
```

Weitere Diskussion auch: diesen Link

SSH Tunnel

TCP Tunnel

Allgemein:

```
ssh -L <lokaler Port>:<Zielrechner>:<Zielport> <Benutzer>@<ssh-Server>
```

Spezialfall X-Weiterleitung:

```
ssh -X <Benutzer>@<Rechner>
```

Dazu muß in der Datei `/etc/ssh/sshd_config` folgender Eintrag vorhanden sein:

```
X11Forwarding yes
```

Tunnel Rückwärts:

```
ssh -R <Port auf ssh-Server>:<Zielrechner>:<Zielport> <Benutzer>@<ssh-
```

```
Server>
```

Dazu muß in der Datei `/etc/ssh/sshd_config` folgender Eintrag vorhanden sein:

```
GatewayPorts yes
```

Tunnel nachträglich anlegen:

```
~C
help
-L <lokaler Port>:<Zielrechner>:<Zielport>
```

SOCKS Tunnel

z.B. für Firefox:

```
ssh -D 1080 <Zielrechner>
```

IP Tunnel / VPN

noch nicht getestet

Client

```
ssh -f -w 0:1 192.168.1.15 true
ifconfig tun0 10.1.1.1 10.1.1.2 netmask 255.255.255.252
route add 10.0.99.0/24 10.1.1.2
```

Server

`/etc/ssh/sshd_config` :

```
PermitTunnel yes
```

```
ifconfig tun1 10.1.1.2 10.1.1.1 netmask 255.255.255.252
route add 10.0.50.0/24 10.1.1.1
```

Links

- SSH Grundlagen von Johannes Franken
- SSH Tunnels von Johannes Franken
- Firewalls durchbohren von Johannes Franken

- SSH FAQ englisch
- SSH Homepage deutsch

1)

Wenn auf dem entfernten Rechner Befehle ausgeführt werden sollen (z.B. Updates einspielen), die nicht unterbrochen werden sollen (z.B. von einem Netzwerkproblem), dann ist screen auf dem entfernten Rechner sehr nützlich

2)

manchmal ist hier auch rsync oder tar nützlich, besonders wenn sowohl die Quell-Datei als auch das Ziel auf entfernten Rechnern liegen

3)

Bei openSuSE 11.0 wird der `ssh-agent` bei der Anmeldung nur gestartet, wenn das Verzeichnis `~/.ssh` existiert:

```
mkdir -m 700 ~/.ssh
```

4)

Mit eingeschaltetem Agent-Forwarding sollte man nur vertrauenswürdige Server besuchen. root-Benutzer auf dem Server können während die Verbindung besteht den ssh-Agent „mitbenutzen“. Dazu sucht root nach

- Prozess-Nummern (`-t`),
- die eine IP-Verbindung geöffnent haben (`-i`),
- deren Namen mit der Zeichenkette `sshd` beginnt (`-c`),
- und die dem verbundenen Benutzer gehören (`-u`)

5)

Weiter sucht er nach dazu passenden Socket-Dateien (`/tmp/ssh-*/agent.$pid`), und speichert eine (`head -n 1`) in der Variablen SSH_AUTH_SOCK.