# Linux am Dienstag

SSH: All Inclusive

# Linux am Dienstag - SSH: All Inclusive

Wofür braucht man die

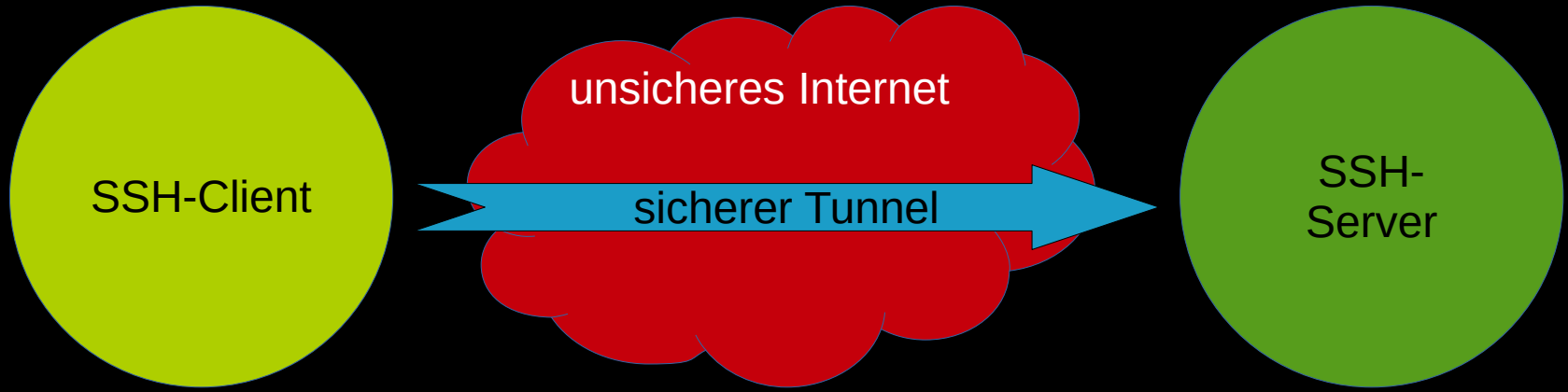
Secure SHell ?

# Linux am Dienstag - SSH: All Inclusive

*„Das Secure Shell (SSH) Protokoll ist ein Protokoll für sicheres Einloggen und andere sichere Netzwerkdienste über ein unsicheres Netzwerk.“*

(RFC 4251)

# Linux am Dienstag - SSH: All Inclusive



# Linux am Dienstag - SSH: All Inclusive

Ursprünglich als Verbesserung von Telnet gedacht,

kann man heute mit SSH sehr, sehr viel mehr erreichen...

# Linux am Dienstag - SSH: All Inclusive

1.

Merke: SSH überträgt Daten nur verschlüsselt!

# Linux am Dienstag - SSH: All Inclusive

2.

SSH kann außer Shellzugriff noch...

# Linux am Dienstag - SSH: All Inclusive

- SSH: Dateien signieren

- SSH: ganze Netzwerke verbinden

- SSH: VPN bauen

- SCP: Dateien direkt vom/zum Ziel kopieren

- SFTP: Dateien direkt vom/zum Ziel kopieren

- SSHFS: entfernte Verzeichnisse mounten
(Hinweis: wird über FUSE realisiert)

# Linux am Dienstag - SSH: All Inclusive

Was braucht man um sich per SSH anzumelden?

# Linux am Dienstag - SSH: All Inclusive

Genau, einen SSH Server ;)

# Linux am Dienstag - SSH: All Inclusive

Den startet man auf dem Zielsystem so:

```
systemctl start sshd
systemctl enable sshd
```

„enable“, damit er beim nächsten Reboot automatisch startet.

# Linux am Dienstag - SSH: All Inclusive

Wie kann man sich per SSH anmelden?

# Linux am Dienstag - SSH: All Inclusive

Entweder per

Username/Passwort

oder

Username/**Schlüssel**

# Linux am Dienstag - SSH: All Inclusive

Das sieht dann so aus:

ssh username@servername

# Linux am Dienstag - SSH: All Inclusive

Per Schlüssel kann man sich so anmelden:

ssh -i pfad/zur/Schlüsseldatei.key username@servername

# Linux am Dienstag - SSH: All Inclusive

oder

# Linux am Dienstag - SSH: All Inclusive

so

ssh username@servername

# Linux am Dienstag - SSH: All Inclusive

# Linux am Dienstag - SSH: All Inclusive

Ähm.. war das nicht vorhin .. schonmal .. da?

# Linux am Dienstag - SSH: All Inclusive

## Der SSH-AGENT

# Linux am Dienstag - SSH: All Inclusive

Der SSH-Agent speichert SSH-Schlüssel sicher

und

übergibt diese auf Anfrage an SSH-Anwendungen.

# Linux am Dienstag - SSH: All Inclusive

Dazu braucht an aber …

# Linux am Dienstag - SSH: All Inclusive

...einen SSH-Schlüssel!

# Linux am Dienstag - SSH: All Inclusive

Diesen Erzeugen wir so:

```
ssh-keygen -t TYPE -f dateiname.key
```

# Linux am Dienstag - SSH: All Inclusive

## Hinweis

Es gibt verschiedene Arten nach denen Schlüssel erzeugt werden können, z.B. RSA und ED25519

# Linux am Dienstag - SSH: All Inclusive

```
$ ssh-keygen -t ed25519 -f Testschlüssel.key -C carolaszugang
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in Testschlüssel.key
Your public key has been saved in Testschlüssel.key.pub
The key fingerprint is:
SHA256:uoBbbJHCPZM2Wxfn4geSWnjftInBy5jZdpEzeOHwvbY carolaszugang
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|                 |
|        o o      |
| .  .  + o O +   |
|  o @  * S @ .   |
|   = @ % X * .   |
|  . B = O * o    |
|   + . o o . .   |
|  .    .    E    |
+----[SHA256]-----+
```

# Linux am Dienstag - SSH: All Inclusive

Wo speichert man den Schlüssel am besten?

# Linux am Dienstag - SSH: All Inclusive

Der *übliche* Ort liegt im Homeverzeichnis des Benutzers:

~/.ssh/

# Linux am Dienstag - SSH: All Inclusive

Der *übliche* Ort liegt im Homeverzeichnis des Benutzers:

~/.ssh/

aber man kann die Datei überall ablegen.

# Linux am Dienstag - SSH: All Inclusive

## Wer startet den SSH-Agenten?

# Linux am Dienstag - SSH: All Inclusive

Das macht Eure Desktop-Environment für Euch,

also z.B. Gnome, Cinnamon oder Plasma

# Linux am Dienstag - SSH: All Inclusive

## Beispiel

| Prozessname | Benutzer | % CPU | Kennun | Speicher | Einheit | Gerätelesevorgänge gesamt |
|---|---|---|---|---|---|---|
| gdm-x-session | marius | 0,00 | 2010 | 516,1 kB | session-2.scope | 143,4 kB |
| cinnamon-session | marius | 0,00 | 2041 | 9,8 MB | session-2.scope | 3,5 GB |
| ssh-agent | marius | 0,00 | 2209 | 98,3 kB | session-2.scope | *N/V* |
| gnome-keyring-daemon | marius | 0,00 | 1989 | 1,6 MB | session-2.scope | 929,8 kB |
| ssh-agent | marius | 0,00 | 4517 | 417,8 kB | session-2.scope | *N/V* |

# Linux am Dienstag - SSH: All Inclusive

Wieso ist das sicher?

# Linux am Dienstag - SSH: All Inclusive

Diese Schlüsselverwaltung hängt an Eurem Login.

Dies bedeutet, dass ohne euer (hoffentlich langes) Loginpasswort, der Schlüsselbund nicht geöffnet werden kann.

# Linux am Dienstag - SSH: All Inclusive

Wie speichert man da jetzt einen Schlüssel?

# Linux am Dienstag - SSH: All Inclusive

Das ist simpel:

ssh-add dateiname.key

# Linux am Dienstag - SSH: All Inclusive

Wenn man nicht möchte,

dass der Schlüssel dauerhaft gespeichert wird,

kann man eine maximale Speicherdauer angeben.

# Linux am Dienstag - SSH: All Inclusive

Für eine Viertelstunde wäre z.B.

ssh-add dateiname.key -t 900

# Linux am Dienstag - SSH: All Inclusive

Wie kann man da reinsehen?

# Linux am Dienstag - SSH: All Inclusive

Entweder im Terminal

ssh-add -l
(kleines L)

# Linux am Dienstag - SSH: All Inclusive

oder mit einer Anwendung wie SEAHORSE.

# Linux am Dienstag - SSH: All Inclusive

Beispiel:

```
$ ssh-add -l
256 SHA256:uoBbbJHCPZM2Wxfn4geSWnjftInBy5jZdpEzeOHwvbY
carolaszugang (ED25519)
```

# Linux am Dienstag - SSH: All Inclusive

Ist der Schlüssel im Agenten hinterlegt,

kann man sich ohne Eingabe eines Passwortes anmelden,

wenn der Zielserver den Schlüssel kennt.

# Linux am Dienstag - SSH: All Inclusive

Moment mal,

woher kennt denn der Zielserver meinen Schlüssel?

Und kann dann nicht jeder auf dem Server mit dem Schlüssel böse Dinge tun?

# Linux am Dienstag - SSH: All Inclusive

1.

Wir haben dem Beispielschlüssel **kein Passwort** mitgegeben,

aber wir könnten und sollten das tun!

# Linux am Dienstag - SSH: All Inclusive

Vorteil:

Kommt die Schlüsseldatei abhanden,
müssen die Finder erstmal das Passwort knacken.

# Linux am Dienstag - SSH: All Inclusive

Wir müssen aber das Passwort nur einmal angeben,
wenn uns SSH-ADD danach fragt,
den Rest erledigt der SSH-AGENT.

# Linux am Dienstag - SSH: All Inclusive

**HINWEIS**

**Wenn der Schlüssel freigeschaltet ist,
kann sich jeder mit Zugriff auf diesen Desktop
auf den vorgesehenen Servern einloggen!**

# Linux am Dienstag - SSH: All Inclusive

**DESWEGEN**

**Die Desktopsession nicht unbewacht offen rumstehen lassen.**

**Die Bildschirmsperre mit Passwort ist verpflichtend!**

# Linux am Dienstag - SSH: All Inclusive

2.

SSH-Schlüssel teilen sich in Private- und Public-Key auf.

# Linux am Dienstag - SSH: All Inclusive

Der Public-Key(Teil) kommt auf den Server.

Damit kann der Server den Schlüssel verifizieren,
der sich da einloggen will.

# Linux am Dienstag - SSH: All Inclusive

Der öffentliche Public-Key sieht in unserem Fall so aus:

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIJrSCG9/iaTDx1Py3pHFx/u2W8jy4rJFHZYOCO4IPQB carolaszugang
```

# Linux am Dienstag - SSH: All Inclusive

Der öffentliche Public-Key sieht in unserem Fall so aus:

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIJrSCG9/iaTDx1Py3pHFx/u2W8jy4rJFHZYOCO4IPQB carolaszugang
```

und da er öffentlich ist, können wir den hier auch abdrucken ;)

# Linux am Dienstag - SSH: All Inclusive

Hinweis:

Das Thema „Asymetrische Verschlüsselung“ wird hier nicht extra behandelt, dazu gibt es andere Vorträge.

# Linux am Dienstag - SSH: All Inclusive

Damit der Login mit einem Benutzer z.B. „root“ mit dem Schlüssel funktioniert, wird der Public-Key auf dem Server in die Datei /root/.ssh/authorized_keys2 eingetragen:

```
# cat /root/.ssh/authorized_keys2
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIJrSCG9/iaTDx1Py3pHFx/u2W8jy4rJFHZYOCO4IPQB carolaszugang
```

# Linux am Dienstag - SSH: All Inclusive

Tada …

```
$ ssh -i Testschlüssel.key root@ziel
Last failed login: Tue Oct 11 09:51:50 CEST 2022 from
31.172.70.163 on ssh:notty
There were 2 failed login attempts since the last successful login.
Last login: Tue Oct 11 09:44:14 2022 from 87.123.110.186
[root ~]# _
```

# Linux am Dienstag - SSH: All Inclusive

OK, wir können einloggen und nu?

# Linux am Dienstag - SSH: All Inclusive

## SSHFS

# Linux am Dienstag - SSH: All Inclusive

mit SSHFS binden ein externes Verzeichnis lokal ein:

sshfs username@zielserver:/home/username/ /mnt

# Linux am Dienstag - SSH: All Inclusive

Damit können wir jetzt auf alle Dateien zugreifen, die

a) in dem Verzeichnisbaum liegen und
b) mit den Rechten des Benutzers erreichbar sind.

Was sehr praktisch ist!

# Linux am Dienstag - SSH: All Inclusive

SSHFS basiert auf SFTP als Protokoll.

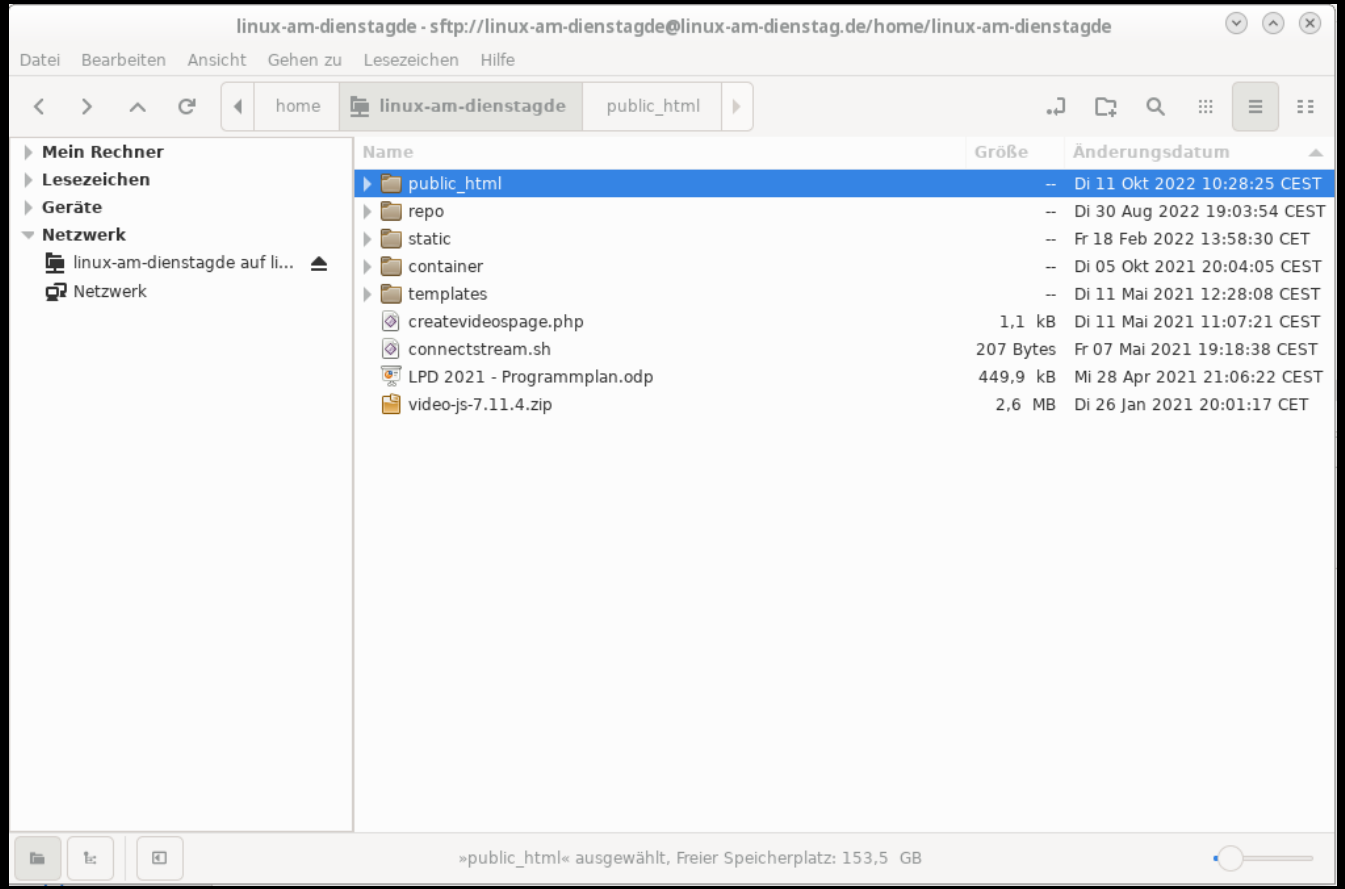
Da ein „Laufwerk“ gemountet wird,
kann man jetzt aber mit jedem Dateimanager auf dieses Laufwerk zugreifen, was den Umgang stark vereinfacht.

# Linux am Dienstag - SSH: All Inclusive

Nemo & Nautilus greifen z.B. per SFTP zu

und stellen den Verzeichnisbaum direkt dar.

# Linux am Dienstag - SSH: All Inclusive



# Linux am Dienstag - SSH: All Inclusive

Eine Alternative zum SSHFS ist SCP.

# Linux am Dienstag - SSH: All Inclusive

Wenn man nur eine begrenzte Anzahl von Dateien kopieren möchte, kann man mit SCP arbeiten:

scp dateiname username@server:/zielpfad/

das geht auch in die andere Richtung:

scp username@server:/quellpfad/dateiname zielpfad/datei

# Linux am Dienstag - SSH: All Inclusive

Trivia:

SCP soll eigentlich von SFTP ersetzt werden,
da beide Protokolle im Endeffekt das gleiche erreichen,
aber nicht alle Distros haben dies schon 1:1 umgesetzt.

# Linux am Dienstag - SSH: All Inclusive

## Die Netzwerkbrücke

# Linux am Dienstag - SSH: All Inclusive

OK, jetzt wird es spannend.

# Linux am Dienstag - SSH: All Inclusive

Der SSH-Server muß passend vorbereitet sein!

# Linux am Dienstag - SSH: All Inclusive

```
[root@server ~]# cat /etc/ssh/sshd_config

Protocol 2

SyslogFacility AUTHPRIV

PasswordAuthentication yes
ChallengeResponseAuthentication no
GatewayPorts clientspecified
PermitTunnel yes

LogLevel info

GSSAPIAuthentication yes
GSSAPICleanupCredentials yes

UsePAM yes
PermitRootLogin without-password

AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL
X11Forwarding no

Subsystem       sftp    /usr/libexec/openssh/sftp-server
```

# Linux am Dienstag - SSH: All Inclusive

„PermitRootLogin without-password“

erlaubt den Root-Login nur mit einem OpenSSH-Schlüssel*

Das verhindert einen Brute-Force-Wörterbuchangriff auf den Rootuser

# Linux am Dienstag - SSH: All Inclusive

„GatewayPorts clientspecified
PermitTunnel yes“

erlaubt es Tunnel auf dem Server zu öffnen.

Das ist wichtig für VPN und Port-Forwarding.

# Linux am Dienstag - SSH: All Inclusive

## SSH-Portforwarding

# Linux am Dienstag - SSH: All Inclusive

Prämisse

Wir möchten auf Heise.de zugreifen,
aber nicht unsere IP zeigen.

# Linux am Dienstag - SSH: All Inclusive

heise.de hat u.a. die IP: 193.99.144.80

```
[root@client ~]# $ ssh -L 4443:193.99.144.80:443 root@server
Last login: Mon Nov  9 11:18:08 2021 from 3.15.1.26
[root@server ~]# $
```

# Linux am Dienstag - SSH: All Inclusive

```
[root@client ~]# $ netstat -lnap|grep 443
(Es konnten nicht alle Prozesse identifiziert werden; Informationen über
nicht-eigene Processe werden nicht angezeigt; Root kann sie anzeigen.)
tcp 0 0 127.0.0.1:4443  0.0.0.0:*     LISTEN   10871/ssh

[root@client ~]# $ curl --insecure  -I https://127.0.0.1:4443/
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Mon, 09 Nov 2021 10:23:52 GMT
Content-Type: text/html; charset=iso-8859-1
Connection: keep-alive
X-Cobbler: servo65.heise.de
X-Pect: The Spanish Inquisition
X-Clacks-Overhead: GNU Terry Pratchett
X-42: DON'T PANIC
Location: https://www.heise.de/
[root@client ~]#
```

# Linux am Dienstag - SSH: All Inclusive

Dem lokalen Klienten PC steht jetzt eine verschlüsselte Verbindung über den SSH-Server zum eigentlichen Ziel zur Verfügung.

# Linux am Dienstag - SSH: All Inclusive

Das Ganze geht auch in die andere Richtung:

SSH-REVERSE-Port-Forwarding

# Linux am Dienstag - SSH: All Inclusive

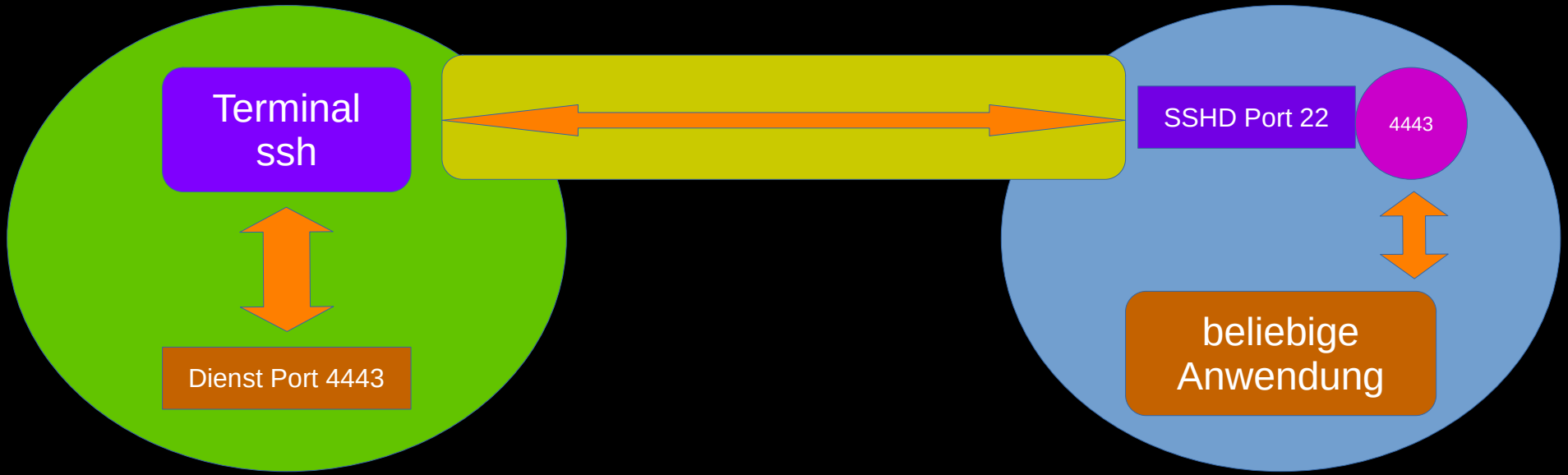
ssh **-R** 4443:zielip:4443 root@Server

Öffne auf dem Server einen Port 4443 und verbinde diesen mit der IP ZielIP im lokalen Netz auf dem Port 4443.

Wenn sich jemand auf dem Server auf Port 4443 verbindet, landet diese Verbindung auf dem lokalen PC.

Die Option „-g“ erlaubt dann den Zugriff auch von außerhalb des Servers.

# Linux am Dienstag - SSH: All Inclusive



# Linux am Dienstag - SSH: All Inclusive

## Die Netzwerkbrücke

# Linux am Dienstag - SSH: All Inclusive

eine Brücke verbindet zwei Netzwerke,
die nicht physikalisch zusammenhängen.

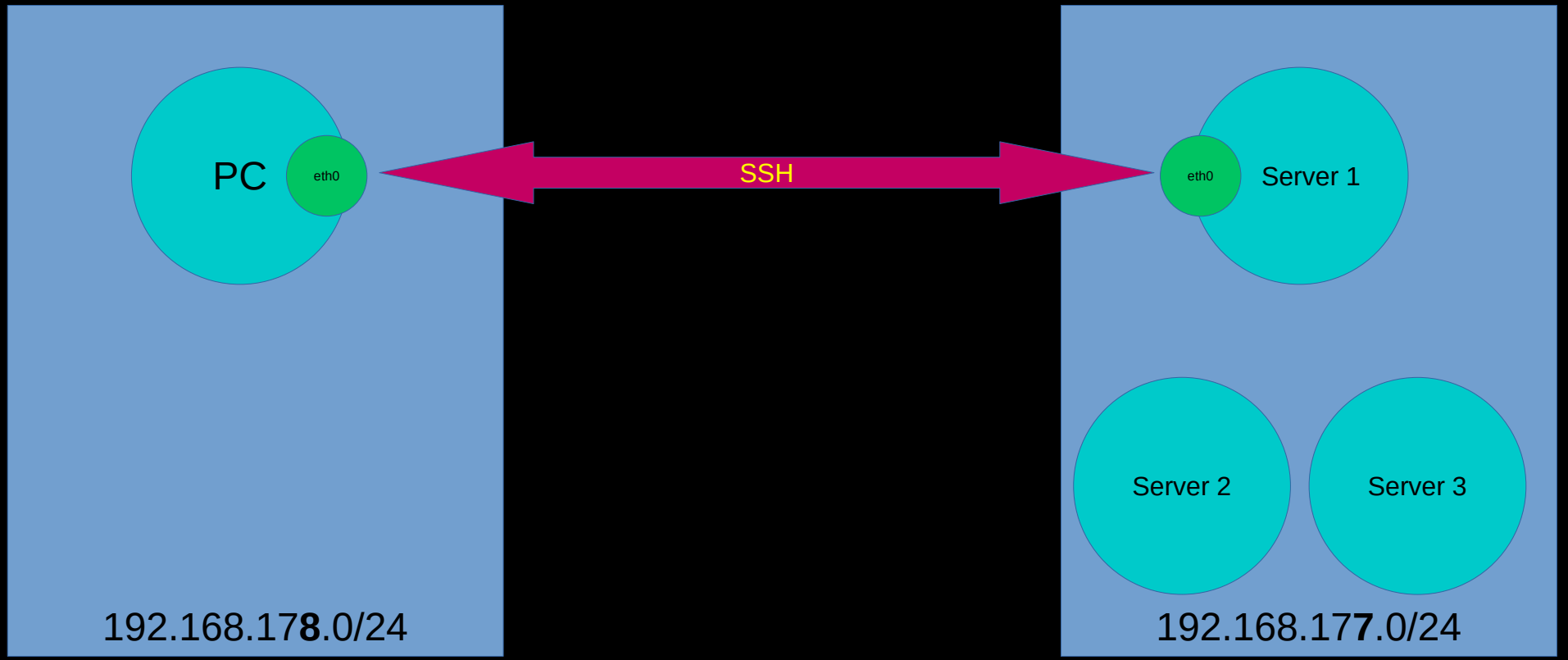
# Linux am Dienstag - SSH: All Inclusive

## Hinweis

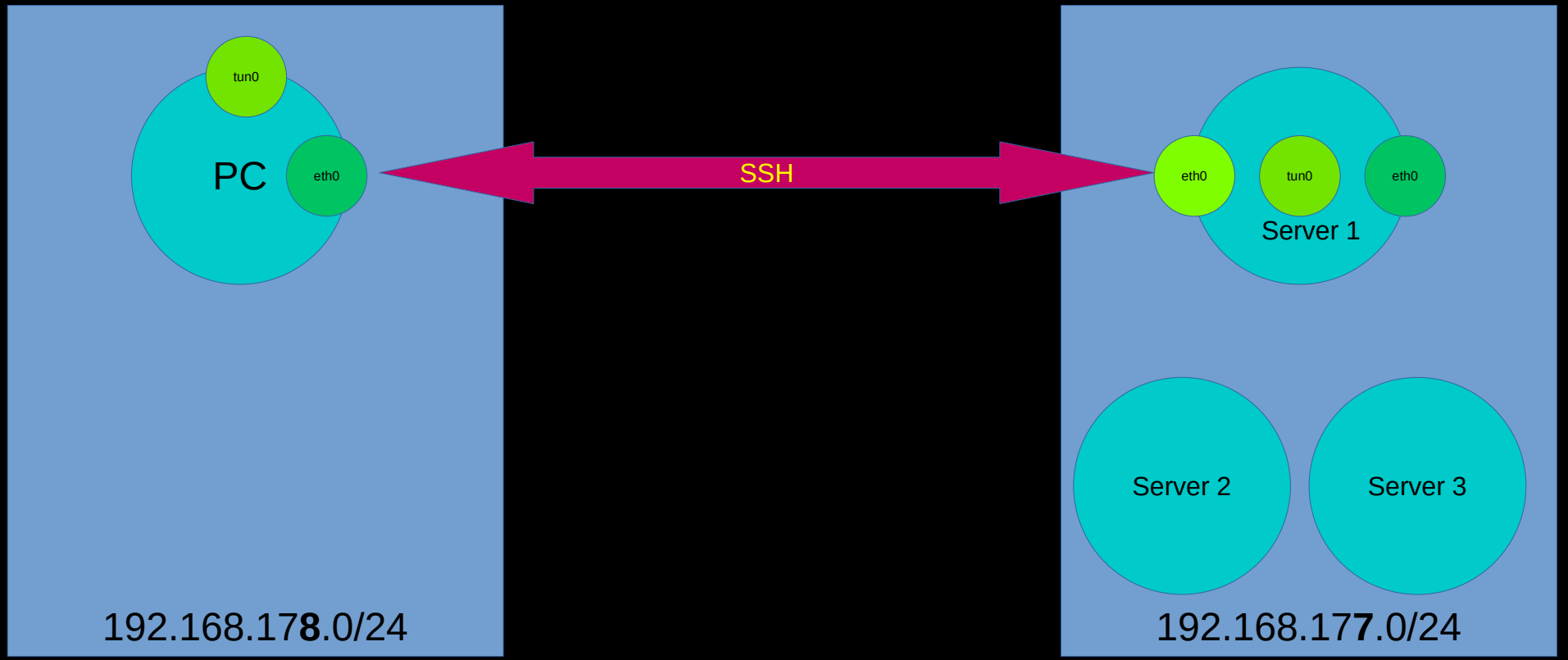
IPv6 ist im nachfolgenden ausgeklammert!
Es handelt sich um Beispielnetzwerke!

# Linux am Dienstag - SSH: All Inclusive

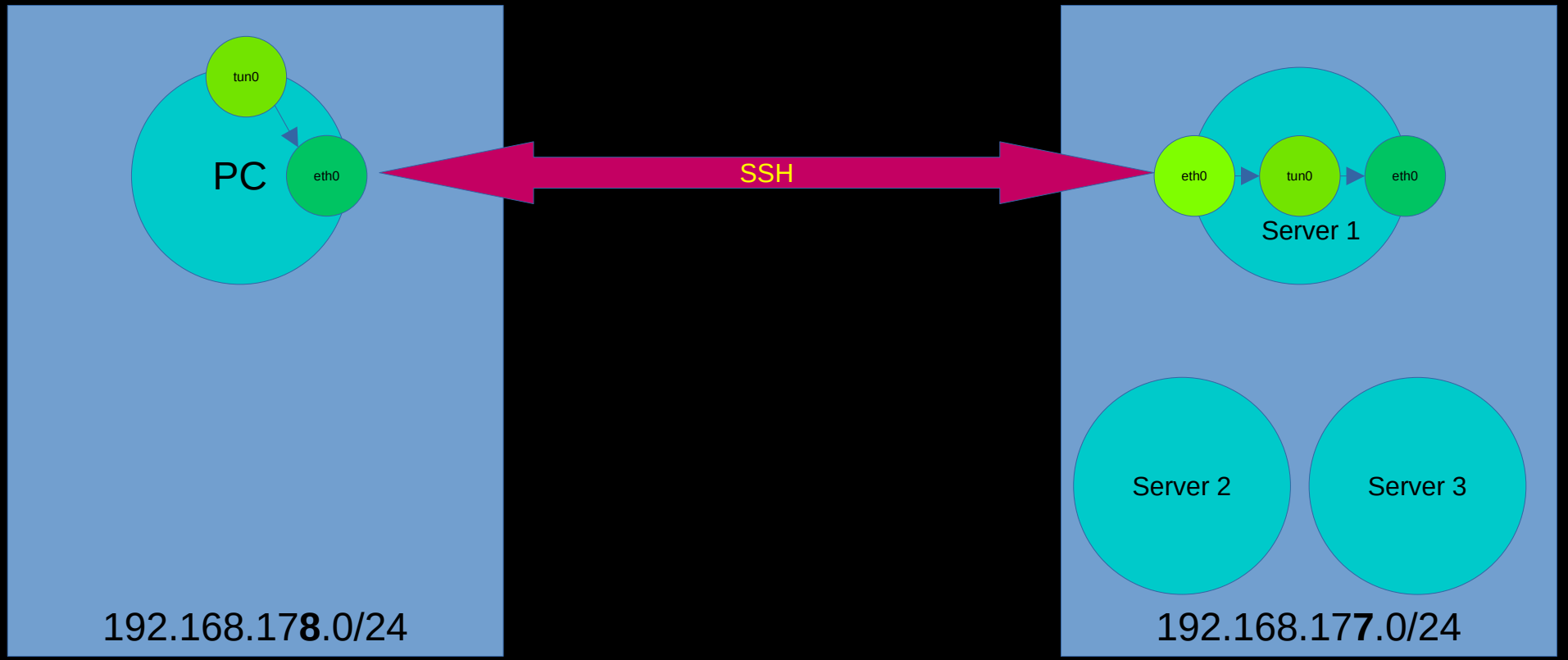
# Linux am Dienstag - SSH: All Inclusive



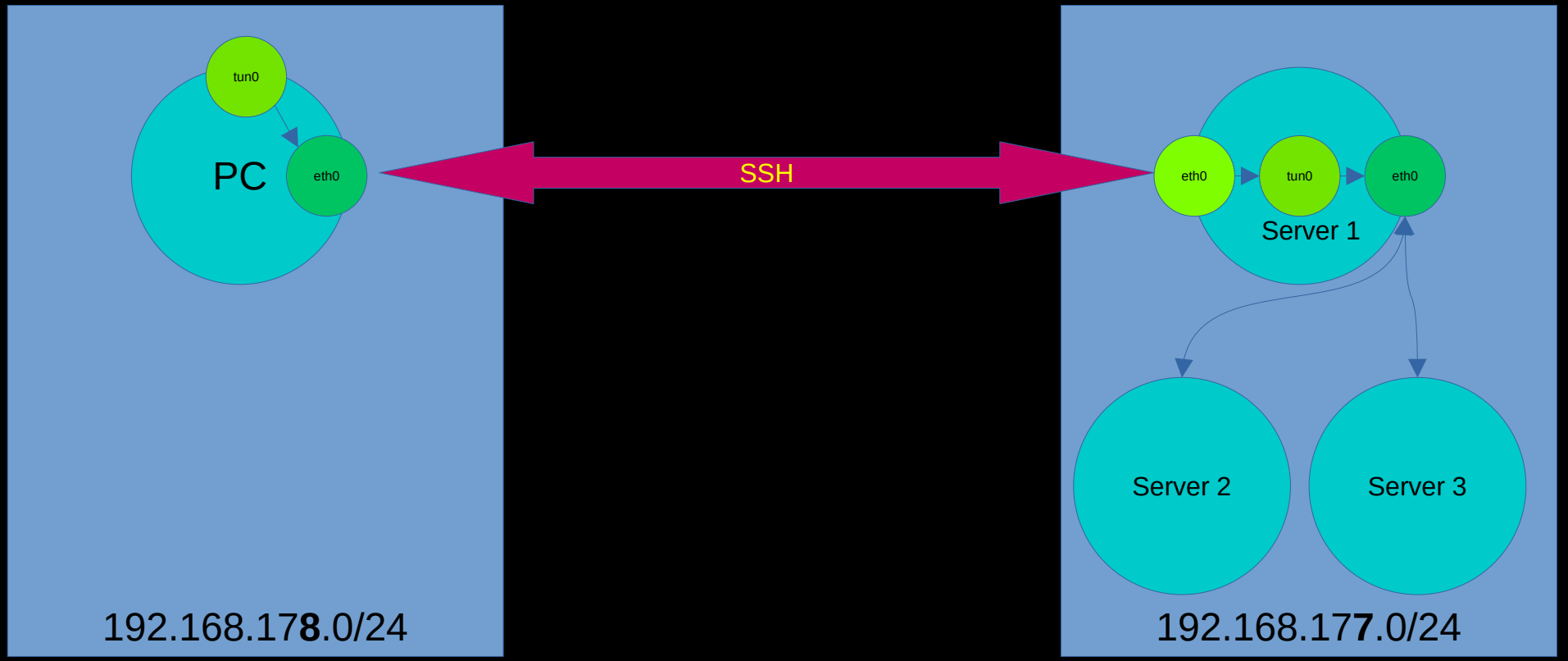
# Linux am Dienstag - SSH: All Inclusive



# Linux am Dienstag - SSH: All Inclusive



# Linux am Dienstag - SSH: All Inclusive



# Linux am Dienstag - SSH: All Inclusive

Die Daten werden über ein virtuelles, von SSH und Kernel zur Verfügung gestelltes,Tunnelinterface (tun0) zum SSH-Server im anderen Netzwerk „tunnelt“.

Von da werden die Daten aus dem Tunnelinterface kommend, über die normale Netzwerkanbindung ins dortige Netz geleitet.

# Linux am Dienstag - SSH: All Inclusive

Um das zuerreichen braucht man definierte Netzwerke und passende Netzwerkrouten.

# Linux am Dienstag - SSH: All Inclusive

Wir brauchen auf Clientenseite...

# Linux am Dienstag - SSH: All Inclusive

## Die Clientseite:

```
[root@client ~]# echo "1" >/proc/sys/net/ipv6/conf/all/disable_ipv6
[root@client ~]# modprobe tun
[root@client ~]# tunctl -t tun0
[root@client ~]# ip link set tun0 up
[root@client ~]# ip addr add 10.0.1.2/32 peer 10.0.1.1 dev tun0
[root@client ~]# route add -net 192.168.177.0/24 dev tun0
```

# Linux am Dienstag - SSH: All Inclusive

GANZ WICHTIG!

```
ssh -NTCf -w 0:0 root@server
```

# Linux am Dienstag - SSH: All Inclusive

“-NTCf“

-N Keinen Befehl ausführen
-T kein Terminal anbinden
-C Kompression aktivieren
-f  SSH in den Hintergrund schicken

“-w 0:0“ benutze Tunnel ID 0 hier und Tunnel ID 0 dort

Damit lassen sich mehrere verschiedene Tunnel zum Server erzeugen.

# Linux am Dienstag - SSH: All Inclusive

## Die Serverseite

```
[root@server ~]# modprobe tun
[root@server ~]# tunctl -t tun0
[root@server ~]# ip link set tun0 up
[root@server ~]# ip addr add 10.0.1.1/32 peer 10.0.1.2 dev tun0
[root@server ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@server ~]# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
[root@server ~]#
```

# Linux am Dienstag - SSH: All Inclusive

„modprobe tun“

Lade TUN Kernelmodul

„ip link set tun0 up“

Fahre ein Netzwerkdevice für Tunnel 0 hoch

„ip addr add 10.0.1.1/32 peer 10.0.1.2 dev tun0“

Setze eine nicht-öffentliche IP und sage dem Tunnel, wer sein Partner ist.

„echo 1 > /proc/sys/net/ipv4/ip_forward“

Erlaube dem Kernel, IPv4-Pakete von einem Interface zum Anderen zu transportieren.

„iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE“

**Aktivieren NAT (Network-Address-Translation), damit IPs aus dem Quellnetz zu IPs des Servers umgesetzt werden. Die Antworten kommen sonst nicht an.**

# Aus der Brücke ein VPN machen

# Linux am Dienstag - SSH: All Inclusive

Technisch gesehen ist eine Brücke schon ein VPN,
weil man Daten durch einen privaten Kanal schickt.

# Linux am Dienstag - SSH: All Inclusive

Im populären Sinn, ist ein VPN eine private Verbindung über ein VPN „Gateway“ ins „freie“ Internet, meint, nicht nur beschränkt auf das Zielnetzwerk.

# Linux am Dienstag - SSH: All Inclusive

Mit einer kleinen Änderung ist das leicht zu erreichen.

# Linux am Dienstag - SSH: All Inclusive

## Die Serverseite bleibt gleich

```
[root@server ~]# modprobe tun
[root@server ~]# tunctl -t tun0
[root@server ~]# ip link set tun0 up
[root@server ~]# ip addr add 10.0.1.1/32 peer 10.0.1.2 dev tun0
[root@server ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@server ~]# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
[root@server ~]#
```

# Linux am Dienstag - SSH: All Inclusive

Die Clientseite ändert sich:

```
[root@client ~]# echo "1" >/proc/sys/net/ipv6/conf/all/disable_ipv6
[root@client ~]# modprobe tun
[root@client ~]# tunctl -t tun0
[root@client ~]# ip link set tun0 up
[root@client ~]# ip addr add 10.0.1.2/32 peer 10.0.1.1 dev tun0
[root@client ~]# route add SSH-SERVERIP gw GATEWAY-IP
[root@client ~]# route del default gw GATEWAY-IP
[root@client ~]# route add default gw 10.0.1.1 dev tun0
```

# Linux am Dienstag - SSH: All Inclusive

Durch die neuen Routen sagen wir unserem Kernel, dass er ALLE Datenpakete, außer denen zum SSH-Server, durch das Tunnelinterface schicken soll, statt direkt über die Netzwerkkarte ins lokale Netz.

# Linux am Dienstag - SSH: All Inclusive

Da die andere Seite für uns NAT macht,
also die eigene IP zur Verfügung stellt,
können die Datenpakete aus dem Netz dort raus,
auch wenn unser Netzwerk 192.168.178.0/24
gar nicht routebar ist.

# Linux am Dienstag - SSH: All Inclusive

Glückwunsch. Überstanden :)

# Linux am Dienstag - SSH: All Inclusive

<HIER, LIEBER LESER, LIVEDEMO DURCHFÜHREN>