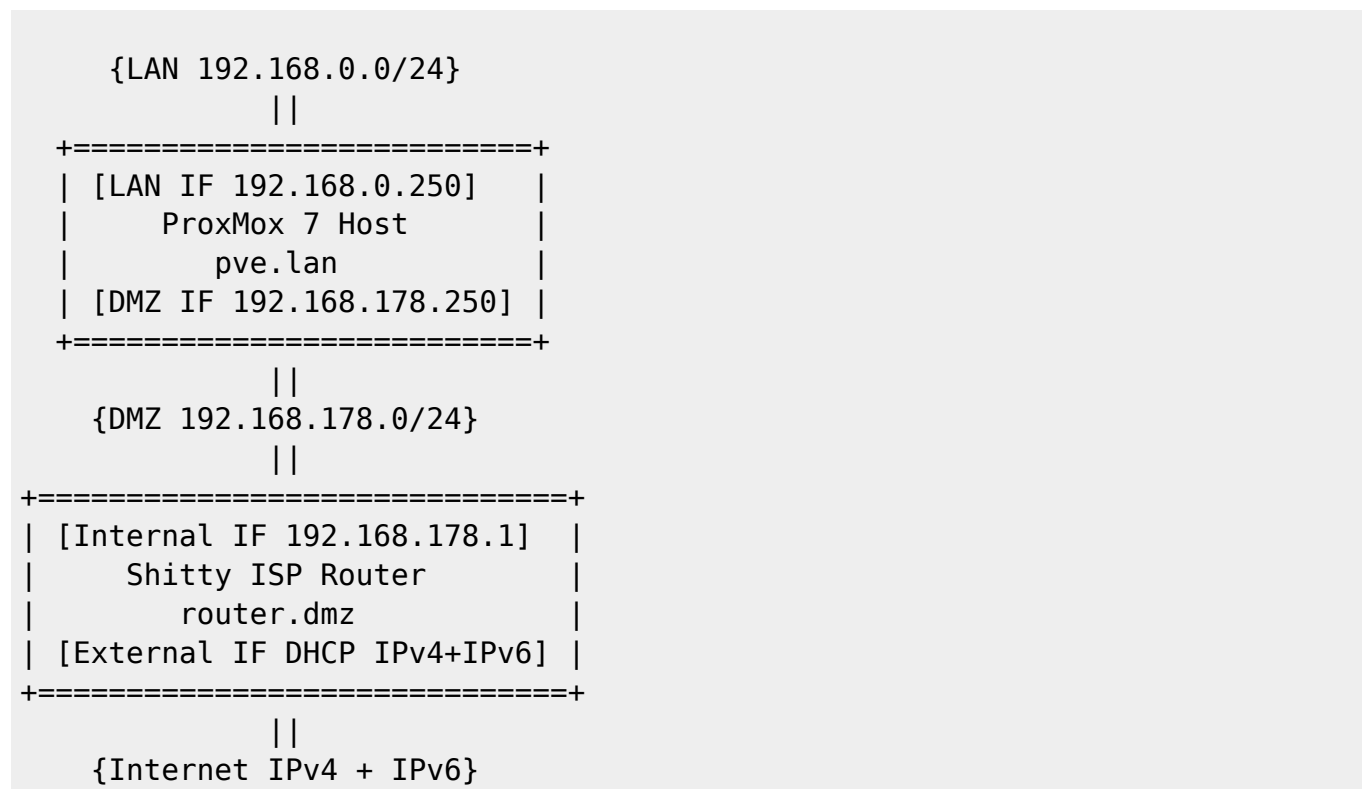





Proxmox 7 Installation and Configuration with two network interfaces

Network Scenario :

This page show the installation notes for a two network interfaced Proxmox 7 virtualisation server. This is my network scenario we are going to work on:



ProxMox 7 Base Installation :

1. Download the latest ProxMox7 ISO to your LINUX workstation.
2. Plug in an empty USB thumb drive larger than the ISO. **Do NOT mount it!**    **All content will be destroyed!**
3. Check out what **base device name** your stick has been assigned. That is the one without any trailing numbers. One may use **lsblk** to list all your blockdevices available to your workstation. Note the device name/path.
4. Write ISO file to the USB stick using **dd**


```

cd Downloads
# NOTE: in my case my thumb drive
# has been assigned the /dev/sdx
# device name.
  
```

```
sudo dd if=/path/to/proxmox7.iso of=/dev/sdx
```

5. Unplug USB Drive and plug it into Proxmox Server. Boot from USB drive. A somewhat graphical Grub Bootmenü should be visible. **Choose install**
6. Follow instructions.
7. When asked for the **management interface** make sure, you choose the “internal” (LAN) interface. Enter a fixed IPv4 address of your LAN, matching netmask and your gateway address if available. **If you do not have an internal gateway just now**, just enter any valid but unused IP address of your LAN by now. Else the PM installer will not let you proceed.
8. Enter an internal FQDN for this PVE host if you like. However this FQDN and its IP address will be used by ProxMox to create and install a “self signed server certificate” on the system, which is used for to protect the traffic to/from the webinterface (TLS/ssl encryption).
9. Proceed and let it install the base System. It should/may reboot automatically after base installation. Remove USB drive. It is no longer needed.
10. After your PVE host is rebooted you should see a login message (issue / motd) on the server console, mentioning the URL to use in the next step to login to the PVE with a webbrowser from any internal (lan) host. So fire up some webbrowser on your workstation:

```
firefox https://pve.lan:8006/
```

11. You should see the webinterface login screen now.  **NOTE: Your browser most likely welcomes you with some security warning. This is because ProxMox uses a “self signed certificate” which is NOT trusted by default by your browser You need to “add a security exception” so it will let you pass and display the website.**
12. Login as “root” with the password you entered at the installation.
13. You now should see the PVE webinterface with a nag screen talking about “not having a subscription”. Ignore that for now. We will switch to the free “non subscription” version soon and dont need a subscription, unless you need enterprise support.
14. At this point in time our PVE system is “not up to date” and still runs the old packages that came with the DVD/ISO. So we should update/upgrade the underlying OS (Debian11) and the PVE Packages as soon as possible for safety reasons!! However in my scenario we don't have “internet access” just yet, since the second networkinterface is still not configured.
15. On the PVE we need to configure the **second (external) networkinterface** now, to gain internet access for the PVE, so we can reach the Debian online repositories. One can use the PVE webinterface for that, or just **go the “Debian way”** of configuring the networkinterfaces by editing the **/etc/network/interfaces** file. This is what i am going for. So first log in to the PVE with SSH and make a backup of the interfaces file:

```
ssh root@pve.lan
# type yes to confirm

# fix the "perl: warning: Setting locale failed" problem
locale-gen

# make a backup first
cp -v /etc/network/interfaces{,.$(date +%F)}
```

16. **NOTE:** Its important to know, that ProxMox always creates “**virtual bridge interfaces**” first and then binds the IP addresses and other network configurations to it. **PVE does NOT configure the physical interfaces (like eth0, enp1s0 etc) directly!** This is required by the

Linux Kernel so it can “share” (bridge) a physical network interface between the kernel (itself) AND containers/VMs. That is why we will add another “bridged interface” for your second (external) NIC in **/etc/network/interfaces**. Use any editor of choice to edit the file (vi, nano, pico are preinstalled). In my case i prefer the “here document” method to **replace** the file with my content:

```
# WARNING: will replace file!

cat << 'EOF' > /etc/network/interfaces
auto lo
iface lo inet loopback

# physical internal LAN interface
iface enpls0 inet manual

# physical external DMZ interface
iface enp5s8 inet manual

# virtial bridged interface 0 (internal LAN)
auto vmbr0
iface vmbr0 inet static
    address 192.168.0.250/24
    bridge-ports enpls0
    bridge-stp off
    bridge-fd 0
    post-up echo 1 > /proc/sys/net/ipv4/ip_forward
    post-up iptables -t nat -A POSTROUTING -s '192.168.0.0/24' -o
vmbr1 -j MASQUERADE
    post-down iptables -t nat -D POSTROUTING -s '192.168.0.0/24' -o
vmbr1 -j MASQUERADE
    post-up iptables -t raw -I PREROUTING -i fwbr+ -j CT --zone 1
    post-down iptables -t raw -D PREROUTING -i fwbr+ -j CT --zone 1

# virtial bridged interface 1 (external DMZ)
auto vmbr1
iface vmbr1 inet static
    address 192.168.178.250/24
    gateway 192.168.178.1
    bridge-ports enp5s8
    bridge-stp off
    bridge-fd 0

EOF
```

17. While the network interfaces are configured now we are still missing the nameserver or DNS information here. Seems the recent **“Debian way”** is still to directly modify the **/etc/resolv.conf** file. So lets make a backup and edit or “replace” this with our version:

```
# make a backup first
```

```
cp -v /etc/resolv.conf{,.$(date +%F)}

# replace /etc/resolv.conf with my config:

cat << 'EOF' > /etc/resolv.conf
search lan
nameserver 2001:4860:4860::8888
nameserver 8.8.8.8
EOF
```

18. The network and therefore internet connection for the PVE system should be configured now. However we still need to fix the “no subscription” thing within the **apt** sources next.
19. Disable their default “subscriber” apt source:

```
sed -i 's/^deb/#deb/' /etc/apt/sources.list.d/pve-enterprise.list
```

20. Enable the “no subscription” apt repository:

```
cat << EOF > /etc/apt/sources.list.d/pve-no-sub.list
# according to ...
# https://pve.proxmox.com/wiki/Package_Repositories
#
# PVE pve-no-subscription repository provided by proxmox.com,
# NOT recommended for production use
deb http://download.proxmox.com/debian/pve bullseye pve-no-subscription
EOF
```

21. Now activate / reload the new network configuration by reboot or by...

```
ifreload -a
# or
ifreload -a -v # VERY VERBOSE
```

22. Now lets update and upgrade all packages for the first time manually:

```
apt update && apt dist-upgrade -yy && reboot
```

23. Repeat all steps for every proxmox node of yours.

Configure Email forwarding :



Since Google decided to drop any support for classic SMTP AUTH (only supports OAUTH2 these days) most tutorials and instructions you can find on the web who still use GMAIL are deprecated and i was unable to find an easy solution to adapt postfix SMTP under Debian 11 (proxmox 7) to it. So i decided to look for another free Email provider (<https://www.sendinblue.com/>) that is still able to serve me plain SMTP AUTH to forward at least “some” notification mails.

This is my attempt to solve this.

It's important to setup automatic email forwarding early on, so the system can notify us on any problem detected. Therefore we entered an email address while we were with the graphical installer. This is ok but it may not be enough to enable proper outbound email.

Since GMAIL failed us and tools like postfix etc are not "oauth2" compliant, i found in Oct.2022 a friendly french email provider called [BREVO](#) that seem to offer plain old **SMTP AUTH** for up to 300 emails per month for free. And for now it seems to accept pretty much any FROM addresses without hassle.

1. Make sure you entered a valid (destination) Email Address while Proxmox Installation. If you need to change it you can do this using the Web UI. Therefore ...
 1. Switch to **SERVER VIEW / DATACENTER / PERMISSIONS / USERS** , then doubleclick on the **root** user to edit
 2. Edit/Change the Email address, then click **OK**
2. Next change the **source email address** that Proxmox uses to send mail FROM. Therefore switch to **SERVER VIEW / DATACENTER / OPTIONS** and double click on **Email from address**
3. Enter a proper **source email address** for your PVE system, that makes easy to identify your source system.
4. Since we need TLS encrypted SMTP AUTH to be used with Postfix SMTP, we need some additional functionality and libraries that go together with the preinstalled postfix MTA, called SASL2. Let's install this:

```
apt install -y libsasl2-modules sasl2-bin swaks

# list all installed/available SASL plugins
saslpluginviewer
```

5. Next we need to configure Postfix and tell it what mailrelay to use and what SASL plugins to use. To edit the main Postfix config we do a backup first:

```
cp -v /etc/postfix/main.cf{,.$(date +%F)}
```

6. Edit some SMTP parameters using **postconf**:

```
postconf 'relayhost = [smtp-relay.sendinblue.com]:587'

postconf 'smtp_use_tls = yes'
postconf 'smtp_tls_security_level = encrypt'
postconf 'smtp_tls_CApath = /etc/ssl/certs/'

postconf 'smtp_sasl_auth_enable = yes'
postconf 'smtp_sasl_security_options ='
postconf 'smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd'
postconf 'smtp_sasl_mechanism_filter ='

systemctl restart postfix
systemctl status postfix
```

7. Store your **SMTP AUTH** login credentials for the mail relay:


```
# NOTE THE LEADING SPACE in the next line!
# Prevents it from being saved in bash history
echo '[smtp-relay.sendinblue.com]:587
axel.werner.1973@gmail.com:seecreepassword' > /etc/postfix/sasl_passwd
chmod 600 /etc/postfix/sasl_passwd

postmap hash:/etc/postfix/sasl_passwd
```

8. You can check the outbound mailqueue with **mailq** to see if the mail is still pending. Or you can watch the logs “live” like this:

```
tail -n0 -f /var/log/messages /var/log/syslog /var/log/mail.* &
```

9. Next we must TEST the mail forwarding and check if our mailbox can receive our Proxmox

notification mails properly.  **MAKE SURE to check the 'SPAM' folder on your mailbox for mails!!!** To send a unique testmail (to root) we can do the following:

```
msg="testies $(date)" ; echo $msg | /usr/bin/pvemailforward
```

Configure Storage Space :

Since my hostsystem does contain multiple terrabytes of diskspace and i would like to use some sort of redundant storage configuration (raid5 a like). ZFS would be a cool choice. However every manual i read notes that i would need at least 1GB of RAM per TB of Disk space, which i dont have. So using ZFS is out of the window by now.

I have 8GB RAM and roughly 3x 6TB HDD (/dev/sd[bcd])

So i guess Linux's 'md' (mdadm raid) it will be.

1. Install the software raid tools:

```
apt install -y mdadm sysstat
```

2. Make sure the drives i want to use are clean from any previous partitioning or filesystems by “wiping” them:

```
wipefs --all --force /dev/sd[bcd]
```

3. Use 'lsblk' to find out more about the disks we like to use. We could just use their classic linux device names like **/dev/sdx** etc, but when it comes to a fault in our future raid it sometimes gets a little stressful if you cant tell for sure which physical drive really is at which SATA port and which drive really failed. Therefor this time i want to use their other drive names instead : **/dev/disk/by-id/*** which contains much more detailed information about each and every drive used. Like vendor, model no, serial no etc. Those information will help you to identify exactly which drive does what. So with this command we can find out now, which drives with their corresponding “id names” are available:

```
lsblk -o PATH,MODEL,SERIAL,STATE,ROTA,TYPE,size | grep disk
```

The MODEL and SERIAL column information is used in the “by-id” names! So make note of those!

4. According to the Arch Linux people its best practice to put a single partition of type 'Linux Raid' on every drive, even it isn't really needed. It's supposed to help later when we'll have to replace a failed drive one day. Also it's easier to see that these disks actually are “in use”. So this is how i create a single partition of type “RAID” with max size on every drive:

```
for drv in /dev/sd[bcd] ; do
    echo "Deploying GPT on ${drv}"
    echo 'label:gpt' | sfdisk -q "${drv}"
    echo "Partitioning ${drv}"
    echo '1M,+,R' | sfdisk -q "${drv}"
    sfdisk -l "${drv}"
done
```

5. Create 3 drive RAID5 array using mdadm :

```
modprobe -a linear multipath raid0 raid1 raid5 raid6 raid10 dm-mod

mdadm --create --run \
    --verbose \
    --level=5 \
    --raid-devices=3 \
    --consistency-policy=ppl \
    --chunk=256 \
    /dev/md/raid5 \
    /dev/sd[bcd]1
```

6. The resync/initialisation of the RAID should have been started and depending on the disk sizes will last several hours or even days. You can check the RAID status by:

```
watch cat /proc/mdstat

# or

mdadm --detail /dev/md/raid5
```

7. However it is not necessary to wait for completion. We can continue on building on it without delay. However things may appear a little more sluggish now, while the raid is still syncing. So now we have a RAID block device named **/dev/md/raid5** now. Let's save it's configuration to **/etc/mdadm/mdadm.conf**

```
mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

8. Test if mdadm's “monitor” service is able to send an alert email (to root, as this is default on proxmox):

```
mdadm --monitor --scan --oneshot --test
```

9. Now let's mark the newly created RAID volume as a "physical" drive for LVM:

```
pvccreate -v /dev/md/raid5

# check details with
pvs

# or

pvdisplay
```

10. What about creating a single LVM group **pvedata** and add the **/dev/md/raid5** as its first physical drive, Therefor we can later grow the VG or "move" it to another drive for easier maintenance.

```
vgcreate pvedata /dev/md/raid5
locale-gen
# check/list with
vgs
# or
vgdisplay
```

11. Since i would like to "share" the whole free disk space between the host os AND Proxmox guests, AND i want to be able to store "anything" (any proxmox content/data type) on this pool i am going to create ONE large logical volume (LV) now and add it to Proxmox as a "directory storage pool" later:

```
lvcreate --verbose --extents 100%VG --name lvraid5 pvedata

# FIXME ?
# lvcreate --autobackup y --extents 100%VG --name lvraid5 --readahead
auto pvedata

# check with

lvs

# or

lvdisplay
```

- 12.



FIXME

- 13.



FIXME

14.



FIXME

15.



FIXME

16.



FIXME

17.



FIXME

18.



FIXME

19. Finally we put some filesystem on the LV:

```
# lookup device path
lsblk -p

# make xfs filesystem
#   - this might take a while
#     on large volumes

# need this for XFS allignment calculations
# take it from your mdadm details
#
export RAID_DEVICE=/dev/mapper/pvedata-lvraid5
export CHUNK_SZ_KB=256
export PARITY_DRIVE_COUNT=1
export NON_PARITY_DRIVE_COUNT=2

mkfs.xfs \
  -L raid5lv \
  -f \
  -l lazy-count=1 \
  -d sunit=$((CHUNK_SZ_KB*2)) \
  -d swidth=$((CHUNK_SZ_KB*2*NON_PARITY_DRIVE_COUNT)) \
  $RAID_DEVICE
```

```
# Check Result / Details:
```

```
xfs_info /dev/mapper/pvedata-lvraid5
```

```
# meta-data=/dev/mapper/pvedata-lvraid5 isize=512    agcount=32,
agsize=91568576 blks
#           =                               sectsz=4096  attr=2, projid32bit=1
#           =                               crc=1        finobt=1, sparse=1,
rmapbt=0
#           =                               reflink=1    bigtime=0
# data      =                               bsize=4096   blocks=2930193408,
imaxpct=5
#           =                               sunit=64     swidth=128 blks
# naming    =version 2                     bsize=4096   ascii-ci=0, ftype=1
# log       =internal log                  bsize=4096   blocks=521728,
version=2
#           =                               sectsz=4096  sunit=1 blks, lazy-
count=1
# realtime  =none                          extsz=4096   blocks=0, rtextents=0
#
```

```
# FIXME
```

```
# mkfs.xfs -L raid5lv /dev/mapper/pvedata-lvraid5
```

```
# Check Result / Details:
```

```
xfs_info /dev/mapper/pvedata-lvraid5
```

```
FIXME
```

20. Prepare a mountpoint on the host and try a manual mount:

```
mkdir -vp /raid5lv
```

```
mount -v /dev/mapper/pvedata-lvraid5 /raid5lv
```

```
mount | grep raid
```

```
ls -la /raid5lv
```

```
touch /raid5lv/Welcome_to_raid5lv
```

```
ls -la /raid5lv
```

```
umount -v /raid5lv  
ls -la /raid5lv
```

21. Add new volume to **/etc/fstab** for automatic mounting on boot:

```
cat << 'EOF' >> /etc/fstab  
LABEL=raid5lv /raid5lv xfs defaults 0 2  
EOF
```

22. Test 'automatic' mounting by:

```
mount | grep raid  
mount -a  
mount | grep raid
```

23. Finally add the mounted directory to the PVE storage manager as storage of type "directory", allowing any data type to be stored on:

```
# creates a SUBDIR proxmox/ so we can share  
# the top level directory with the host  
# or other things without mixup.  
#  
pvesm add dir raid5lv --path /raid5lv/proxmox/ --content  
iso,vztmpl,backup,rootdir,images,snippets
```

24. From this moment on you should be able to find this new storage area within the web UI and add/upload VM and container templates, ISOs and all that jazz to it, as well as creating VMs and containers. But that's a topic for another section.
25. For some additional cleanup (removing default **data** LV and resizing the **rootfs** i did the following:

```
# FIXME  
lvremove /dev/pve/data  
  
#  
lvresize --resizefs --extents +100%FREE /dev/pve/root
```

26. Now there should be much more free space available in the **rootfs** (more than 30GB) than before:

```
df -h / /raid*
```

27. Finally (after the raid is finished syncing) let's do a little performance test (with pvw on board tools) and compare the ssd boot device with the lvm on mdadm raid:

```
pveperf # root fs  
  
pveperf /raid5lv/
```

```
CPU BOGOMIPS:      19956.04
REGEX/SECOND:      1060547
HD SIZE:            109.80 GB (/dev/mapper/pve-root)
BUFFERED READS:    268.40 MB/sec
AVERAGE SEEK TIME: 0.14 ms
FSYNCS/SECOND:     480.64
DNS EXT:            536.55 ms
DNS INT:            539.24 ms (lan)
```

[illegible]

For whatever reason the company behind ProxMox changed the **.bashrc** of the ROOT user. Debian's bashrc template is **/etc/skel/.bashrc** , which contains important environment variables to configure the bash history. However they are missing in ROOT's version, resulting in dupes and possibly security threatening entries.

- Extract and append to ROOT's bashrc:

1. logout and login

In theory the mdadm monitor “should” already send us email to the **root** account, in case of some

problem with the MD raid. However i prefer to add another layer of security to it and **make it audible** when there is a state detected which is not expected.

Some simple beeps are sufficient. But sending an S.O.S. by morse code is more like it. However it can easily be extended to do more things (like sending another mail) or sending a whole message with details via morse code. It's up to you.

This is how i did it:

1. Install the "beep" tool:

```
apt install beep
```

2. Load/Add Linux Kernel Module (driver) for the PC speaker:

```
modprobe pcspkr
```

3. Test beep :

```
beep
```

4. Now we make sure that this Kernel Module is loaded on every reboot automatically. So reboot and repeat the test if beep still works without "modprobing".
5. Now you can do lots of things. Add it to your shell scripts or cronjobs. Whatever you prefer. Here is a little demo on how i watch the RAID health status (alias md raid, mdadm) and use the PC speaker to alert me while daytime by MORSE CODE. :
6. Prepare the shell script which tests the RAID health and does the morse:

[/usr/local/sbin/check-raid-status.sh](#)

```
#!/bin/bash

# CHANGE LOG:
#
# 2021-08-01 A.Werner   ADD: wall + console output + new string
#                  clean,checking
#
# 2021-09-05 A.Werner   ADD: new OK string added
#

function dit {
    beep -f 2750 -l 75 -d 50
}

function dah {
    beep -f 2750 -l 175 -d 50
}

function spc {
    sleep .1
}
```

```
function s {
    dit
    dit
    dit
    spc
}

function o {
    dah
    dah
    dah
    spc
}

function morse_sos {
    s
    o
    s
    sleep .5
}

mdState=$( /usr/sbin/mdadm --detail /dev/md127 | grep "State :" |
cut -d: -f2 | tr -d ' ' )

case "$mdState" in
    active|active,checking|clean|clean,checking)
        : # nop
        ;;
    *)
        morse_sos
        echo "$0 WARNING: mdadm reports md0 status: '$mdState' on
$(date)" >&2
        echo "$0 WARNING: mdadm reports md0 status: '$mdState' on
$(date)" >/dev/console
        wall "$0 WARNING: mdadm reports md0 status: '$mdState' on
$(date)"
        ;;
esac
```

7. Make shellscript executable:

```
chmod +x /usr/local/sbin/check-raid-status.sh
```

8. Activate a cronjob that runs the check script every minute while daytime:

```
cat <<'EOF' > /etc/cron.d/raid-monitor-md127
#
# Regular cron jobs to audibly alert admin if
# md (mdadm) raid changes state from "clean"
# using morse code
```

```
#  
#  
#m h dom mon dow user  command  
5 06-22 * * * root /usr/local/sbin/check-raid-  
status.sh  
  
EOF
```

9. If this does not work for you or is too sensitive it should be no problem to pimp it up relatively easy.

FIXME :

1.



FIXME

2.



FIXME

3.



FIXME

4.



FIXME

5.



FIXME

6.



FIXME

7.



FIXME

8.



FIXME

9.



FIXME

From:
<https://awerner.myhome-server.de/> - Axel Werner's OPEN SOURCE Knowledge Base

Permanent link:
<https://awerner.myhome-server.de/doku.php?id=it-artikel:linux:proxmox-7-installation-and-configuration-with-two-network-interfaces&rev=1687626128>

Last update: 2023-06-24 17:02

