An EndeavourOS Simple Home Server on x64_86.

For security reasons, this server is not intended to be accessed from the internet and should be connected directly by LAN cable to a router or to a switch connected to the router. If you need remote access for some of your files, a Cloud based service would be a better choice for those files. IMHO a wireless server is NOT a good idea at all, avoid it if possible.

This tutorial will set up a "headless" simple home LAN file server observing the KISS principle. Keep It Simple & Secure. It requires the server computer to have a monitor, keyboard, and mouse temporarily attached during installation. After installation the monitor, keyboard, and mouse can be removed, and the server can be run headless and administered from a Gnu/Linux Desktop Client.

Storage devices can be a SSD, Hard Drive or USB Drive. This guide installs a base EndeavourOS operating system on one storage device.   Data is stored on it's own separate storage device, and a third storage device is used for DATA backup. Since most servers are on line 24 X 7, the ideal hardware for this home server would be low power. However, you can use just about any hardware you have laying around.

INSTALL BASE EndeavourOS

ON THE SERVER COMPUTER.
Install a storage device on the SATA 1 connector on the motherboard to receive the Endeavour OS. This device needs to be at least 32 Gb or larger.
WARNING:: ANYTHING ON THIS STORAGE DEVICE WILL BE DESTROYED.

Insert the latest EndeavourOS ISO thumb drive into a USB port.
For a server, I prefer a msdos/MBR boot if the computer allows it. I think that MBR is more problem free than an UEFI install. You can choose which ever you desire.
Boot the computer. It is recommended to get into BIOS and disable WiFi if possible.
During bootup, Press F11 or whatever is appropriate for your computer, and select msdos/MBR or UEFI/GPT for your desired method of installation.

After the EndeavourOS ISO boots up, if you know the IP address of your router, you can skip the following instructions in the box and go to the next page.

open a terminal window.

```
$ ip addr
   2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> YaDa YaDa
      inet 192.168.0.103/24 brd 192.168.0.255 scope global dynamic noprefixroute enp3s0
```

enp3s0 is the ethernet interface name, 192.168.0.103 is the IP address, /24 is the netmask, and dynamic indicates the IP address was assigned by DHCP and is dynamic.

On a piece of paper, record your ethernet interface name, in this case enp3s0 (that is a zero on the end). Your interface name may be different such as eth0 or enp8s0 or whatever.
Record the IP address 192.168.0.103. Now we know the the first three triads of the router's IP address in this case 192.168.0 or it could be different such as 192.168.1 The router's full IP address usually ends with 1 such as 192.168.0.1 or 192.168.1.1 but let's verify

```
$ sudo pacman -S nmap
$ sudo nmap -sn 192.168.0.0/24      (the first 3 triads of your router plus .0/24 to scan entire router)
  Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-14 20:05 UTC
  Nmap scan report for 192.168.0.1
  Host is up (0.00024s latency).
  Mac Address: 86:36:F9:87:3A:4B (Tp-link Technologies)
SNIP
   Nmap done: 256 IP addresses (8 hosts up) scanned in 1.79 seconds
$
```

I know my router is a Tp-link, so out of 8 hosts I only showed the router's IP address 192.168.0.1
close the terminal window

"START THE INSTALLER".
At the "Please choose an option for installation" choose "Online (choose your desktop)"
Install as usual until you get to "Package selection".

At "Package selection":
"Base-devel + common packages" should be the only thing checked. Leave it as such.
Open up "Base-devel + common packages" and UNcheck everything except the following:

```
base-devel     yay     linux-headers     linux-firmware   netctl     net-tools
efibootmgr     dosfstools     mtools    openssh     intel-ucode     amd-ucode
wget     bash-completion
```

For “Users”
  What is your name? Public Share
  What name do you want to use to log in? pshare   (pshare makes the tutorial easier)
  What is the name of this computer? enosServer   (or whatever name you wish)
  Choose a password to keep your account safe. Enter Password: Re-enter Password:
  do NOT select Log in automatically or Use the same password for administrator account.
  Choose a password for the administrator. Enter Root Password Re-enter Root Password

After installation, do not enable "reboot now", then Click "Done" and power down.
Remove the installation USB drive. And reboot. Set BIOS to boot off SSD if necessary.

After boot up, you should get something similar to this

```
Arch Linux 5.4.7-arch1-1 (tty1)

enosServer login:
```

Enter "root" as the username
followed by the root password you chose at installation.

You should get the # prompt indicating root as follows.
[root@enosServer ~]#

During boot up, some computers will not run Grub properly with this stripped down version of EndeavourOS. If the up and down arrows don't work, and pressing e doesn't enter the edit mode then do the following after logging in as root. Otherwise skip the commands in the box.

```
# vi or nano  /etc/default/grub
   GRUB_GFXPAYLOAD_LINUX=keep    change  keep to text
   GRUB_THEME=/boot/grub/themes/endeavouros/theme.txt   add a leading # to comment out
# grub-mkconfig -o /boot/grub/grub.cfg
# reboot
```

Create a Static IP address

The install does not configure our network, this must be done manually.
Servers work much better with a static IP address rather than using a dynamic IP address with DHCP. We need to set a static IP address for this server.
List the network device.

```
   # ip link
    snip
    2: enp3s0: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state DOWN .......
    snip
```

Look for line 2 to determine the ethernet device name, in this case enp3s0. It possibly could be eth0 or enp8s0 or others. I believe the device name is determined by hardware, e.g. what NIC chip is on the motherboard.

configure and enable the network device.

```
   # cd /etc/netctl/examples
   # ls
      ethernet-static  ....plus several more
   # cp ethernet-static /etc/netctl
   # cd /etc/netctl
   # vi or nano /etc/netctl/ethernet-static
     Description='a basic static internet connection'     (leave as is)
     Interface=enp3s0                                     (enter your network device)
     Connection=ethernet                                  (leave as is)
     IP=static                                            (leave as is)
     Address=('192.168.0.150/24')                         (enter desired static IP, I like 150)
     #Routes=('192.168.0.1/24 via 192.168.0.1')           (leave commented out)
    Gateway='192.168.0.1'                                 (adjust IP for your router)
    DNS=('192.168.0.1' '8.8.8.8')                        (most routers have DNS pass through)
                                                         (so use the Gateway IP for DNS)
                                                     ( 8.8.8.8 Google DNS as secondary DNS)

   # netctl start ethernet-static
   # netclt enable ethernet-static
   # ip addr
     2: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> YaDa YaDa
     inet 192.168.0.150/24 brd 192.168.0.255 scope global enp8s0
```

Notice that the IP address should be what you entered. You now have a static IP address.
Check your connection

```
   # ping -c 6 endeavouros.com
```

Should get good results with 0% packet loss
Reboot and rerun the ping command to ensure the network auto started at boot up.

CONFIGURE EndeavourOS SERVER

```
# date
  Thu 23 Jan 2020 01:50:52 PM MST     (check time and date are correct)
```

Some might prefer to run the LTS kernel on their server. If so:

```
  # pacman -S linux-lts  linux-lts-headers
  # pacman -R linux  linux-headers
  # grub-mkconfig -o /boot/grub/grub.cfg
  # reboot
```

Clean up a few unused libraries and such.

```
# pacman -Rnsdd $(pacman -Q | grep libx | awk '{print $1}')
# pacman -Rnsdd xterm jfsutils reiserfsprogs thin-provisioning-tools mdadm dhcpcd xfsprogs
lvm2
# pacman -Syu
# pacman -Qdt
```

pacman -Qdt shouldn't show any orphans.
At this point the base EndeavourOS installation is at 1.9 GB with 167 packages.
An Arch base install the Arch way is 1.9GB with 150 packages. A diff of 17 packages.
Eleven of the extra packages are endeavouros related or quite useful:

```
endeavouros-keyring     endeavouros-mirrorlist     grub2-theme-endeavouros
git    inet-utils     linux-headers    logrotate   man-db  man-pages  net-tools   yay
```

So, there are really only 6 packages that would be considered bloat. It is not practical to try and eliminate them. This is about as close to an Arch base install as you can get.

Make changes to user pshare

The user pshare needs to be added to the group users

```
# gpasswd -a pshare users
# groups pshare
  sys log network floppy scanner power adm wheel audio lp optical storage video users rfkill
  pshare
```

For security reasons, the user pshare should be a member of as few groups as possible.
Remove pshare from every group except users and pshare as follows.

```
# gpasswd -d pshare sys            (removing pshare from group sys)
   pshare has now been removed from the group sys.
```

Press the up arrow to bring up the last command, then backspace out “sys” and enter “log” in it's place, press Enter   Repeat this for the following groups.

network floppy scanner power adm wheel audio lp optical storage video rfkill
enter the groups command again and it should list just users and pshare

```
# groups pshare
   users pshare
```

This is probably overkill for a home LAN server.   And yes, this does eliminate sudo from user pshare. IMHO a server is no place for sudo. Do all your administrative duties as root.

Add the alias ll for ls -l command

I prefer the alias ll (List Long) installed as an alias for ls -l   EndeavourOS does not do this. If you want to use ll do the following with vi or nano

```
# vi or nano /etc/bash.bashrc
    alias ll='ls -l   - -color=auto'            (add this line at the end of the file)
```

Log out by typing in “exit” and log back in to activate the new alias. Enter ll and it should complete without any error. If you made a typing error while entering the alias, bash will report "command not found".

Check Hostname

You set the hostname for your server during install. If you wish to change it

```
# hostnamectl status
   Static hostname:  enosServer.localdomain
   YaDa YaDa
```

The “Static hostname” is the only line of interest.   It probably says
Static hostname: enosServer   (if that is what you set at installation)
If you don’t like enosServer, do the following to change the hostname

```
# hostnamectl set-hostname MyServer.localdomain  - -static
```

Of course you can use any host name you want, but the .localdomain part is necessary.
Rerun

```
# hostnamectl status           (to verify your new hostname if you changed it.)
   Static hostname:  MyServer.localdomain
```

Explore the system we just installed

Enter

```
# df -h    (df is Display Filesystem, -h uses M and G for filesizes which is more human friendly)
locate /  and check the install size under "Used".   For me, 1.9 GB.

# pacman -Q > enosPkgs
# wc -l enosPkgs
   167 enosPkgs            ( 167 is the number of installed packages)
# more enosPkgs             (to look at the package list)
```

SETUP THE SSH SERVER

To run the server headless, SSH has to be set up so a Linux client can access the server. For security reasons, DO NOT use the default SSH port 22. There are 65,000 plus port numbers available. Go to Wikipedia's TCP port list or other such list. Scroll down to the 9000 – 10,000 range. Some of these ports are not assigned to any thing. Choose one of these ports to use for your SSH Local Area Network. As root, edit sshd_config file and change the entries listed. Note: for the last two items listed, simply remove the leading # to uncomment the option.

```
# vi or nano /etc/ssh/sshd_config
            FROM #Port 22   TO  Port 9XXX          (whatever port you choose)
            From   #PermitRootLogin prohibit-password   TO   PermitRootLogin no
            From #PasswordAuthentication yes  TO  PasswordAuthentication  yes
            From #PermitEmptyPasswords no  TO   PermitEmptyPasswords no

# systemctl disable sshd.service        (disable old ssh using port 22)
# systemctl enable sshd.service         ( make sure a symlink was created)
# systemctl start sshd.service
# systemctl status sshd.service        (Should be Active(running) with no alerts or failures.)
```

SETUP A FIREWALL

You can never be TOO security conscientious. Time to set up a firewall.

```
# pacman -S ufw                         (install Uncomplicated Fire Wall)
# ufw status                            (check status of ufw)
    status: inactive
# ufw logging off       (otherwise logging appears on the screen & makes a mess of DMESG)
# ufw default deny
# ufw allow from 192.168.0.0/24 to any port 9XXX   (9XXX is your chosen ssh port)
# ufw enable
    Firewall is active and enabled on system startup
 # ufw status
 Status: active
    To                  Action        From
    --                   ------        ----
   9XXX               ALLOW      192.168.0.0/24

# systemctl enable ufw.service
# systemctl start ufw.service
# reboot
```

After reboot, login as root then

```
# ufw status
 Status: active
    To                  Action        From
    --                   ------        ----
   9XXX               ALLOW      192.168.0.0/24
```

To ensure firewall was activated at boot up.

In summary, we have installed ufw, and enabled systemd to start the ufw service at boot up. We enabled the firewall itself, and set the default action to deny. Then we set the following rule.

```
ufw allow from 192.168.0.0/24 to any port 9XXX
```

Now anything coming in from any IP address on our private LAN (192.168.0.0/24) going to the ssh port (9XXX) is allowed. Anything coming in from an IP address outside of our private LAN is rejected. In other words, the entire world is blocked but any computer on our ethernet LAN is accepted. Since I have personal stuff on this server, I don’t want anyone from outside my house to have access to it. If you want internet access to your data, use a cloud service.

INSTALLING A DATA SSD

Power off the computer and install a SSD on SATA port 2, or connect a USB 3 external enclosure with a SSD or 3.5 inch hard drive installed. Boot up the computer. Login as root

```
# lsblk
NAME   MAJ:MIN RM   SIZE   RO TYPE MOUNTPOINT
sda         8:0       0  119.2G  0   disk
├─sda1    8:1       0       1G   0    part    /boot
├─sda2    8:2       0  110.4G  0    part     /
└─sda3    8:3       0      7.9G  0    part     [SWAP]
sdb         8:16      1       30G  0    disk
└─sdb1    8:17      1       30G  0    part
```

It can be seen that /dev/sda1 /dev/sda2 and /dev/sda3 are the partitions for our OS device. Another clue is sdb1 does not have a Mount Point. The newly added device is /dev/sdb1 If the device is brand new and has never been partitioned it may look different. Now that it is determined that /dev/sdb is our DATA device

```
# fdisk /dev/sdb
Command  o                    (that's lower case o -- create a new empty DOS partition table)
Command  n                                  (add a new partition)
   Partition type: p                        (p = primary)
  partition number: 1
  First sector:  enter to accept default
  Last sector: enter to accept default
   Partition #1 contains a vfat signature.        (this warning may not appear, if so yes)
   do you want to remove the signature?  yes
Command: w                                  (write table to disk and exit)
```

```
# mkfs.ext4 -L DATA /dev/sdb1
```

We need to manually set up the /server mount point. As root

```
    # cd /                          (Change to root directory)
    # mkdir /server
    # chown root:users /server
    # chmod 744 /server
```

You should now have something similar to this snippet.

```
# ll
   drwxrwxr- -  46 root users 4096 Aug 15 22:15 server
```

The /server directory is used for mounting SSD partitions. You should never put any files or sub-directories in this reserved directory.

Modify /etc/fstab

Find the UUID that was assigned at formatting for the DATA SSD partition.

```
# blkid
   /dev/sda1: UUID="d026ab30-1a28-4e18-8bca-6b07b05a03c9" TYPE="ext4"
   /dev/sda2: UUID="2dbbf1ae-d7b8-4209-8265-89fcccc6cdac" TYPE="ext4"
   /dev/sda3: UUID="136842e6-89c9-4f97-9dca-70067fdd1d98" TYPE="swap"
   /dev/sdb1: LABEL="DATA" UUID="b4dc7162-fcde-4b28-b8b9-e98626932902" TYPE="ext4"
```

you should see /dev/sdb1 with a nice label of "DATA" and its UUID number.
Copy the UUID number on a sheet of paper without the quotes

```
# cp /etc/fstab /etc/fstab-bkup     (always make a back up of config files before editing)
```

Using vi or nano, add the following line at the end of the /etc/fstab file

```
UUID=Your-UUID-Number  /server  ext4   defaults,relatime,discard  0 2
```

close /etc/fstab

```
# lsblk
NAME   MAJ:MIN RM   SIZE   RO TYPE MOUNTPOINT
sda         8:0      0   119.2G  0    disk
├─sda1    8:1      0        1G   0    part    /boot
├─sda2    8:2      0   110.4G  0    part     /
└─sda3    8:3      0      7.9G  0    part     [SWAP]
sdb         8:16     1       30G  0    disk
└─sdb1    8:17     1       30G  0    part
```

sdb1 should show no mount point, if it does show mounted, # umount /dev/sdb1

```
# mount -a                  (to test if /etc/fstab is correct after our editing)
# lkblk
NAME   MAJ:MIN RM   SIZE   RO TYPE MOUNTPOINT
sda         8:0      0   119.2G  0    disk
├─sda1    8:1      0        1G   0    part    /boot
├─sda2    8:2      0   110.4G  0    part     /
└─sda3    8:3      0      7.9G  0    part     [SWAP]
sdb         8:16     1       30G  0    disk
└─sdb1    8:17     1       30G  0    part     /server
```

If sdb1 now shows a mount point of /server then fstab is configured correctly.
Reboot the server computer.

If it boots up normally, Login as root and skip the box.

If the computer takes a long time to boot up, you made a typo and it can't find the SSD. After it times out, It will say:

```
You are in emergency mode.
Blah Blah
Give root password for maintenance
(or press Control-D to continue):
```

Type in your root password and do a blkid to check the UUID & device name such as /dev/sdb
Edit /etc/fstab and look for typos. When you find your mistake, reboot and see what happens.

Make sure the /server directory is correct. You should have something similar to this snippet.

```
# ll /
  drwxrwxr--   3 root users   4096 Dec 28 11:15  server
```

check the permissions (drwxrwxr- -) and ownership (root users) and make sure they are correct If they are correct, then fstab is doing its job and skip the box. If they are not correct, then follow the instructions in the box

```
# cd /
# chown root:users /server
# chmod 774 /server
# ll /
  drwxrwxr--   3 root users   4096 Dec 28 11:15  server
```

Reboot the computer, then then login and become root

```
# ll /
```

Recheck the permissions (drwxrwxrt- -) and ownership (root users) and make sure they are correct

If you edit the /etc/fstab file for any reason, after reboot be sure to check /server for the permissions and ownership again as editing fstab has a nasty habit of changing stuff when mounting devices to /

If they are not correct, issue the following and recheck:

```
   # chown root:users /server
   # chmod 774 /server
```

Now that ownership and permissions are set
```
    # su pshare       (Switch User to pshare)
    $ ll /server
     drwx------ 2 root root 16384 Dec 28 11:15  lost+found
```

You should see the lost+found directory created during formatting, at least with ext4
Now test and make sure pshare can access /server
```
$ echo "this is a test" > /server/test
$ ll /server
   -rw-r--r-- 1 pshare pshare     15 Dec  9 19:21  test
$ cat /server/test
  this is a test
$ exit
# htop                    (out of curiosity check memory usage, 2 GB is sufficient)
   81.2M/1.86G
```

On it’s own separate SSD you have a working partition at /server for all your data. Always work in /server as a user. lost+found was generated by the computer as root. Which is fine as it is only for the computer's use. Everything else in /server should belong to user pshare, including all files and directories.

Your EndeavourOS server is now complete. The monitor, keyboard, and mouse can be removed to run the server headless. Maintenance can be performed in a Linux Client using SSH.

To make this modular, four more installments will follow. The users can pick and choose want they want to install in their LAN server, These will include:
1. Configure a Linux to Linux client
2. Install SAMBA in the server and configure a Windows client.
3. Install minidlna in the server and setup minidlna in a Linux or Windows client.
4. Last, but definitely not least, format an external USB 3 storage device and prepare for doing backups of the Data disk. This is an absolute must.

As part of KISS (Keep It Simple & Secure) all the packages used for doing all this are in the regular Arch repositories. No third party software, or AUR packages thus far. That is KISS to the utmost. In case you want to play with something in the AUR such as Plex Media Server or Kodi, base-devel, linux-headers or linux-lts-headers, and yay were included during installation.

Also part of KISS, the network is controlled by netctl, a systemd based network configuration and start up. netctl is about as KISS as you can get.