Install a LAN Server on an ARM SBC

August 2, 2020

Objective: Install a minimal Operating System on an ARM SBC (Single Board Computer) for use as a simple home LAN server. The server will utilize a micro SD card or eMMC card for the Operating System, one USB 3.0 storage device for the server's DATA, and one USB 3.0 storage device for DATA backup.

Select a port number for SSH

For security reasons, do not use the default SSH port 22. Think of this as a PIN. Instead of Personal Identification Number it is a Port Identification Number. Pick a four or five digit PIN. Go to wikipedia's TCP port list and choose a port number between 8000 and 48000.

TCP and UDP port numbers
https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Unused numbers are not listed. Look for a group of numbers that aren't being used, such as 9339 is listed (used) and the next number listed is 9389. That's a block of 50 port numbers that you can use. Choose something in the middle. If you want a 5 digit PIN scroll to 10,000 and above and look for unused blocks.

9339	Yes		Clash of Clans, a mobile freemium strategy video game	Unofficial
9389	Yes	Yes	adws, Microsoft AD DS Web Services, Powershell uses this port	Official

The script will prompt you for your chosen SSH port number.

The instructions use host name = enosServer and user name = pshare (public share).
If you want to follow the instructions implicitly, use this hostname and username.
They can changed to something more appropriate for your use, but be sure to allow for this while following the instructions.

Other information needed:
Desired hostname: enosServer
Desired user name: pshare
Desired Last Octet of a Static IP address for this computer
Desired SSH port number as chosen above
root password
user password

As mentioned above, the SBC server will store your data on an external USB 3.0 SSD enclosure. The option of letting the script automatically partition, format, and mount the USB 3 enclosure will be offered. To utilize this, a USB 3.0 SSD enclosure with a SSD of your choice and capacity, will need to be connected to the SBC before running the script.

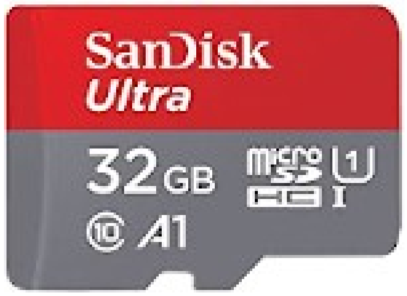
The benefits of an ARM SBC device for a headless LAN file server is it's small size and very low power consumption.

Theoretically you can use any ARM device that Arch Linux Arm supports. Just go to the Arch Linux Arm site, https://archlinuxarm.org/ hover over "Platforms" then the Architecture, brand name, etc. to see if your device is supported. Click on your device and get a page with overview and installation tabs. The installation tab has instructions for installing an Arch Linux ARM image on a micro SD card or eMMC card if your SBC offers eMMC.

In our testing we used an Odroid N2, an Odroid XU4, and a Raspberry PI 4b as the test beds. For running a headless server, all three performed well. I am sure there are others available, but these are the three that were available for testing. The main requirements for a good reliable server are, a 1 Gbit ethernet port, and two USB 3.0 connectors for the external SSDs.

Flash the micro SD card with the OS and bootloader.

The storage device for the Operating System will be either a Micro SD card or an eMMC card. The micro SD card must be at least 16GB, a 32GB gives more head room for logs and etc. Use a good name brand micro SD such as Samsung or SanDisk ultra, not an off brand. SD cards come in different speed classes, 2, 4, 6, and 10, with 10 being the fastest. Since the advent of 4k devices, there is also UHS class speed 1 and UHS class speed 3. UHS = Ultra High Speed. Get at least a class 10 HC1 device. Here is what to look for.



The speed class is the number 10 inside what looks like a C.
The speed class is followed by A1 which is a new specification I am not familiar with.
The UHS speed is the number 1 inside a U and also referred to as HC1. This one sold for $8.98 USD at amazon including a Micro SD to SD adapter.
The Odroid N2 and Odroid XU4 will also run off of eMMC

On a working x86_64 computer, insert the latest USB EndeavourOS ISO installer. Use the USB EndeavourOS ISO because it includes GParted, and cleaning up directories and files created during the flash process is not necessary as the ISO is not persistent.

Boot into the msdos/MBR version of the EndeavourOS installer ISO.
Insert the USB card reader containing the Micro SD to SD adapter with the micro SD card.
In the Endeavour welcome screen, select Partition Manager (Gparted)

INSTALL Base ARCH LINUX ARM on an ODROID XU4

Click on "GParted" tab and select the USB SD READER   (ensure the right device is selected)
Note the Device Name of the SD READER, such as /dev/sdb. Write this down.

Click on "Device" tab, and create a msdos Partition Table.
  If an existing partition is mounted, this will fail. Highlight the partition with the key symbol, right click on it, select “unmount” then try again.
Click on "Partition" tab, then new

```
Free Space preceding MiB:   1          Create as:   Primary Partition
New Size MiB:   leave as is            Partition name:
Free Space following (MiB):   0        File System:    ext4
Align to:   MiB                        Label:  Odroid-XU4
```

Apply All Operations
Close GParted

Close the Welcome screen and open a terminal window.
Some of the following commands can take several minutes, so be patient.

```
$ mkdir Odroid
$ cd Odroid                (isolates the process from the rest of the OS)
$ sudo su
# mkdir MP                 (MP = temporary Mount Point for the USB SD Reader)
# mount /dev/sdX1 MP    (change /dev/sdX1 as noted in Gparted, e.g. /dev/sdb1)
# wget http://os.archlinuxarm.org/os/ArchLinuxARM-odroid-xu3-latest.tar.gz  NOTE: 1
# bsdtar -xpf ArchLinuxARM-odroid-xu3-latest.tar.gz -C MP
# cd  MP/boot
# ls -l                            ( should see the file sd_fusing.sh )
# sh sd_fusing.sh /dev/sdX      (change /dev/sdX to what's appropriate. e.g. /dev/sdb)
# cd ../..
# umount MP       (this could take a while)
# exit
$ exit
```

The Arch Linux ARM Operating System is now installed. NOTE: 2

Shut down the x86_64 computer, and remove the uSD card from the USB SD Reader.
Set the boot switch selector on the Odroid-XU4 board, next to the HDMI jack, to the uSD position (to the left). Insert the micro SD card into the XU4. Connect a monitor, keyboard, ethernet, and if you want auto partitioning, formatting, and mounting of the external DATA SSD, connect a USB 3.0 SSD enclosure to the Odroid-XU4. Apply 5 VDC.

NOTE 1 Notice the URL starts with http and not https, and the URL has xu3 in the filename.
The XU-3 and XU-4 use the same SOC (System On a Chip) so they share the same image.

NOTE 2 To install OS on an emmc card, see "Flash an emmc card" at the end of this tutorial.

INSTALL Base ARCH LINUX ARM on an ODROID N2

Click on "GParted" tab and select the USB SD READER   (ensure the right device is selected)
Note the Device Name of the SD READER, such as /dev/sdb. Write this down.

Click on "Device" tab, and create a msdos Partition Table.
If an existing partition is mounted, this will fail. Highlight the partition with the key symbol, right click on it, select “unmount” then try again.
Click on "Partition" tab, then new

```
   Free Space preceding MiB:   1            Create as:   Primary Partition
   New Size MiB:   256                      Partition name:
   Free Space following (MiB):  XXXX        File System:  fat32
   Align to:   MiB                          Label:  BOOT
```

Create a second partition

```
   Free Space preceding MiB:   0            Create as:   Primary Partition
   New Size MiB:   XXXXXX                   Partition name:
   Free Space following (MiB):  0           File System:  ext4
   Align to:   MiB                          Label:  ROOT
```

Apply All Operations
Close GParted

Close the Welcome screen and open a terminal window.
Some of these commands can take several minutes, so be patient.

```
   $ mkdir Odroid
   $ cd Odroid                (isolates the process from the rest of the OS)
   $ sudo su
   # mkdir MPboot             (MP = temporary Mount Point for the USB SD Reader)
   # mkdir MProot
   # mount /dev/sdX1 MPboot    (change /dev/sdX1 as noted in Gparted, e.g. /dev/sdb1)
   # mount /dev/sdX2  MProot
   # wget http://os.archlinuxarm.org/os/ArchLinuxARM-odroid-n2-latest.tar.gz   NOTE: 1
   # bsdtar -xpf ArchLinuxARM-odroid-n2-latest.tar.gz  -C MProot
   # mv MProot/boot/*   MPboot
   # dd if=MPboot/u-boot.bin  of=/dev/sdX conv=fsync,notrunc  bs=512 seek=1
   *** For eMMC Edit  MProot/etc/fstab    change boot partition to /dev/mmcblk0p1 ***
   # umount MPboot   MProot
   # exit
   $ exit
```

The Arch Linux ARM Operating System is now installed.
Shut down the x86_64 computer, and remove the uSD card from the USB SD Reader.
Set the boot switch selector on the Odroid-N2 board to MMC. Insert the micro SD or eMMC card into the N2. Connect a monitor, keyboard, ethernet, and if you want auto partitioning, formatting, and mounting of the external DATA SSD, connect a USB 3.0 SSD enclosure to the Odroid-N2. Apply 5 VDC.

NOTE 1 Notice the URL starts with http and not https.

INSTALL Base ARCH LINUX ARM on a Raspberry Pi 4b

Click on "GParted" tab and select the USB SD READER   (ensure the right device is selected)
Note the Device Name of the SD READER, such as /dev/sdb. Write this down.

Click on "Device" tab, and create a msdos Partition Table.
If an existing partition is mounted, this will fail. Highlight the partition with the key symbol, right click on it, select “unmount” then try again.
Click on "Partition" tab, then new

Free Space preceding MiB: 2	Create as: Primary Partition
New Size MiB: 150	Partition name:
Free Space following (MiB): XXXX	File System: fat32
Align to: MiB	Label: BOOT

Create a second partition

Free Space preceding MiB: 0	Create as: Primary Partition
New Size MiB: XXXXXX	Partition name:
Free Space following (MiB): 0	File System: ext4
Align to: MiB	Label: ROOT

Apply All Operations
Close GParted

Close the Welcome screen and open a terminal window.
Some of these commands can take several minutes, so be patient.

```
$ mkdir Rpi4
$ cd Rpi4                  (isolates the process from the rest of the OS)
$ sudo su
# mkdir MPboot             (MP = temporary Mount Point for the USB SD Reader)
# mkdir MProot
# mount /dev/sdX1 MPboot    (change /dev/sdX1 as noted in Gparted, e.g. /dev/sdb1)
# mount /dev/sdX2  MProot
# wget http://os.archlinuxarm.org/os/ArchLinuxARM-rpi-4-latest.tar.gz   see NOTE 1
# bsdtar -xpf ArchLinuxARM-rpi-4-latest.tar.gz  -C Mproot      see NOTE 2
# sync
# mv MProot/boot/*   MPboot
# umount MPboot   MProot
# exit    then   $ exit
```

The Arch Linux ARM Operating System is now installed. Shut down the x86_64 computer, and remove the uSD card from the USB SD Reader. Insert the micro SD card into the Rpi4. Connect a keyboard, ethernet, a monitor to HDMI 0 (next to USB Type C power connector), and optional external USB 3 DATA SSD enclosure. Apply 5 VDC.

NOTE 1: The URL starts with http and not https, for the Official Archlinux Arm 32 bit image. For an unofficial 64 bit Raspberry Pi 4b image, download the latest image from:
https://olegtown.pw/Public/ArchLinuxArm/RPi4/rootfs/   into the Rpi4 folder you made.
Pro: It’s 64 bit & it works. Con: trust factor plus kernel and firmware updates are manual.
NOTE 2: If using moonman’s 64 bit image, substitute that filename in the bsdtar command.

Install EndeavourOS on ARM

The following SHOULD work with any ARM SBC (Single Board Computer) once Arch Linux ARM base is installed on the device.

The default user is *alarm* with the password *alarm,* the default root password is *root.*
FYI, alarm = Arch Linux ARM and it is also the default hostname.

After the SBC boots up:
Login as root   (enter root for the username and root for the password)

```
  # use vi or nano to edit /etc/pacman.d/mirrorlist
     comment out the server under "## Geo-IP based mirror selection and load balancing"
     un-comment servers near you
   # pacman-key --init
   # pacman-key --populate archlinuxarm
   # pacman -Syy
   # pacman -Syu
   # pacman -S git
   # systemctl reboot   & Login as root
```

RUN SCRIPT TO INSTALL Base OS

```
# git clone https://github.com/endeavouros-arm/install-script.git
# cd install-script
```

Make endeavour-ARM-install-V1.0.sh executable

```
# chmod 774 endeavour-ARM-install-V1.0.sh
# ls -l
  -rwxrwxr--  1  root root  5607  May   5  01:39  endeavour-ARM-install-V1.0.sh
# sh  endeavour-ARM-install-V1.0.sh   (run the installer script)
```

Choose to install a headless server environment. The following procedures are performed.

Additional packages will be installed to complete the server environment.
Enter the requested information required for configuration.
The time zone is set
NTP is enabled
The hardware clock is synced if present.
Locale is set
Hostname is set
/etc/hosts is configured
Enter root password
delete default user "alarm" & create specified user name
Enter user password
Configure for the specified static IP
Configure SSH including specified port number
Configure and enable the ufw firewall
Provides the option to auto partition, format, and mount the USB 3.0 SSD for DATA

The script installs three alias's which you might see in the instructions.

```
alias la='ls  -al  - -color=auto'   I think of it as "ls all" including dot files
alias lb='lsblk -o NAME,FSTYPE,FSSIZE,LABEL,MOUNTPOINT'   think "ls blks" (partitions)
alias ll='ls  -l  - -color=auto'  I think of as "ls long".
```

If the external USB SSD enclosure was formatted, partitioned, and mounted by the script, then you now have an installed and fully configured headless server that is ready to go. We should ensure that the install went well, and this will also get you familiar with your server.

POST INSTALL

Now is the time to do a little exploring and see if everything is as expected.
log in as root using your new root password.

```
# ping -c 4 endeavouros.com          (should get 0% packet loss)
# ip addr                            (line 2: inet should show static IP you entered)
# date
  Thu 23 Jan 2020 01:50:52 PM MST     (check time and date are correct)
# df -h            the root directory  will show approximately 1.9 Gbyte used for the install
# pacman -Q > pkglist
# wc -l pkglist             (will show approx 177 packages installed)
  177 pkglist
# less pkglist         (will display installed packages)
# htop                 ( will display memory usage)
```

If you had the script prepare the USB 3 external DATA SSD for you, there should be two directores named /server and /serverbkup. These are the mount points for the DATA device and the DATABKUP device when doing backups. Check the permissions and ownership for these two directories as shown below, which are edited for clarity

```
# ls -l /                (listing of the root directory)
drwxrwxr--.  root users   server
drwxrwxr--.  root users   serverbkup
```

If they are not correct, issue the following commands as root and recheck

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

```
# ufw status
  Status: active
     To                    Action        From
     --                    ------        ----
   9830                 ALLOW      192.168.0.0/24
```

Now anything coming in from any IP address on our private LAN (192.168.0.0/24) going to the ssh port (XXXX) is allowed. The entire world is blocked, except any computer on your ethernet LAN is accepted on port XXXX.

If the USB 3 DATA device was prepared during installation, it's time to check it out.

```
# ls -al  /server
   drwx------ 2 root root 4096 Jun 24 11:51  lost+found
                              (directory lost+found was created with mkfs.ext4 – we don't need it)
#  rm -rf  /server/lost+found

# su pshare                    (Switch User to pshare, or your choice of username. see NOTE:)
$ echo "This is a test" > /server/test
$ ls -al /server
   -rw-r—r--  1  pshare pshare  15 Jun 24 14:02  test
$ cat /server/test
   This is a test
$ exit                ( then exit twice to log out of enosServer back to the Console window tty)
# exit
```

NOTE: On it's own separate SSD you have a working partition at /server for all your data. Always work in /server as a user. Everything in /server should belong to user pshare, including all files and all directories.

You are now finished with your simple Home LAN server install.   The monitor and keyboard can be removed to run the server headless. Maintenance can be performed in a Linux Client using SSH. If you chose NOT to prepare the DATA device during install, instructions to manually prepare it are on page 15 of the instructions. Please prepare the USB 3 external DATA SSD before attempting to add any data to the server.

After your first install, as a shortcut you can highlight the actual commands used, and skip all the explanations.

At this point, the server is completely installed and configured. Time to set up a Linux Client to utilize the server.

SET UP A LINUX CLIENT COMPUTER

On a CLIENT LINUX COMPUTER with XFCE or other desk top environment, open a Terminal window. Use the static IP address you established for the server and the host name you decided on. Add the following line to the end of /etc/hosts. This will establish the route to the Server. Your info may be different from the example.

```
$ sudo vi or leafpad  /etc/hosts      (add to end of file)
    192.168.0.150    enosServer.localdomain      enosServer
```

This establishes a relationship between the host name of enosServer and it's IP address, kind of like a DNS service. Otherwise you would have to address the server by it's IP address and port number, which can be awkward and hard to remember.

Since the SSH port that the enosServer listens on was changed from the default port 22, the Client computer needs to know which port to use.

```
$ sudo vi or leafpad /etc/ssh/ssh_config              (change the following line.)
      FROM  #Port 22        TO   Port 9XXX      (Use the same port # as in the server)
```

The SERVER's firewall was setup during installation to accept requests from any CLIENT computer on our LAN at Port 9XXX.

CONFIGURE THE CLIENT FIREWALL

If you already have a firewall installed, or you don’t want a firewall, skip this section.

```
$ su                          (Then enter your root password)
# pacman -S gufw                              (install ufw firewall)
# ufw status                                 (check status of ufw)
  Status: inactive
# ufw logging off             (otherwise logging appears on screen & makes a mess of DMESG)
# ufw default deny            ( denies all incoming UNSOLICITED traffic, solicited traffic passes)
# ufw enable
# systemctl enable ufw.service
# systemctl start ufw.service
# systemctl reboot                          (then log back in as user)
$ sudo ufw status
  Status: active
```

Connect to enosServer with SSH

In a terminal window, as a user try to connect to enosServer

```
 $ ssh pshare@enosServer        (if you happen to be at root, dont try to SSH as root)
```

If you successfully communicated with enosServer, you will get something similar to:

```
   The authenticity of host '[enosserver]:9XXX ([192.168.0.XXX]:9XXX)' can't be established.
   ECDSA key fingerprint is 54:fa:20:25:c1:91:d3:3d:4c:8c:47:02:32:f2:5e:8e.
   Are you sure you want to continue connecting (yes/no)?
```

The above dialog is a one time thing. Type in “yes”, the connection should be completed and you will be asked for pshare's password. You should then have a terminal prompt of [pshare@enosServer ~]$

You are now logged into enosServer as user pshare. Pay attention to the prompt, it will always let you know which computer you are in and the user name. The SSH server on enosServer was configured to NOT allow login as root. If you need to use root for administration, type in su and enter your server's root password. Anything you type in the Terminal window will now be executed in the enosServer computer. You can execute pacman -Syu or perform other administrative routines.

Type in

```
# exit        (if you are in as root)
$ exit        (exit again as user.  This is the proper way to log out of a network connection.)
```

You should be back to your client computers prompt.

On the LINUX CLIENT COMPUTER, do not terminate your enosServer session by clicking on the X (close) button at the top right of your terminal window. After properly disconnecting from the enosServer you should return to your local prompt in the terminal window. Now you can close your Client side terminal window by typing in exit once more.

SSH into enosServer and bypass the password

This is all and good, but it is a pain to have to enter the password all the time. Now ssh keys may need to be generated in our client computer as user.

```
$ ll .ssh
    -rw-------. 1 don don 1831 Jun  8  2019 id_rsa
    -rw-r--r--. 1 don don  406 Jun  8  2019 id_rsa.pub
```

If the ll command produces the above results, you already have ssh keys. You can skip the instructions in the box. If you don't have ssh keys, enter the following commands

```
$ ssh-keygen -t rsa -b 2048   (create  ssh keys for user, hit enter 3 times for defaults )
$ ls -al                      (  .ssh should be drwx------  if not  $ chmod 700 .ssh  then recheck)
   drwx------  don don  4096 Jan  5  1309  .ssh
$ ll .ssh
        -rw-------. 1 don don 1831 Jun  8  2019 id_rsa
        -rw-r--r--. 1 don don   406 Jun  8  2019 id_rsa.pub
```

The id_rsa file is your PRIVATE SSH Key and you should never do anything with it. Don't copy, move, or otherwise mess with it. Just leave it alone. The id_rsa.pub file is your PUBLIC SSH Key. We need to export your PUBLIC SSH Key to the enosServer.

```
 As user
$ ssh-copy-id  -i  ~/.ssh/id_rsa.pub  pshare@enosServer
   enter pshare's password when requested.
```

You will be returned to your client's prompt. Now try to log into enosServer

```
$ ssh pshare@enosServer
```

You should now be logged into enosServer as pshare without having to enter your password. To do administration, just ssh into the Server from a client terminal window Once in enosServer change to root and do most anything you want from a nice GUI terminal window with mouse, scroll bars, cut and paste, etc. The server and client are now configured to remotely administer the server.

```
[pshare@enosServer ~]$ exit           repeat exit until back in the client computer.
```

Use FUSE and SSHFS to view Server data in a file manager

Now we will set up FUSE and sshfs to use a Thunar window to access the files on the server.
IN THE CLIENT COMPUTER as user install sshfs.

```
$ sudo pacman -S sshfs
```

IN THE CLIENT COMPUTER, in a Terminal window as user

```
$ mkdir  /home/$USER/enosServer        (make a mount point for the enosServer)
```

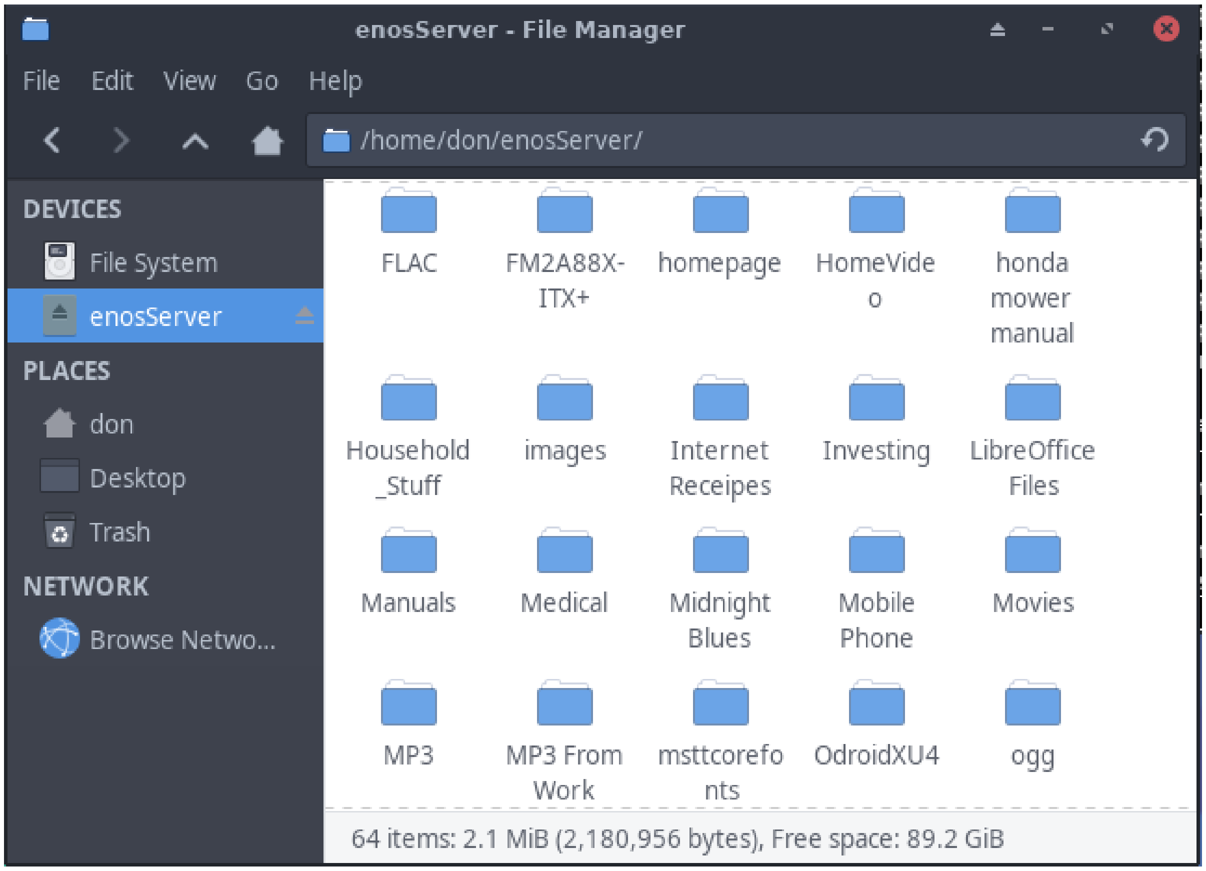
The /home/$USER/enosServer directory is a Mount Point for the enosServer. You should never add any directories or files locally in this directory. If enosServer is not mounted, the enosServer directory should always be empty.

Here is the part you've been waiting for. Still in the CLIENT LINUX COMPUTER, as user

```
$ sshfs pshare@enosServer:/server  /home/$USER/enosServer
$
```

It should complete without any errors.

Launch Thunar, then Click on Home and click on enosServer. You should see the file "test" that was created when the DATA SSD was added.
Ideally, the enosServer folder should have nothing but sub folders to organize your data. Eventually as the enosServer folder is filled up, it could look something like this.



64 items: 2.1 MiB (2,180,956 bytes), Free space: 89.2 GiB

Also notice in the left column under the DEVICES, the enosServer Icon. Click on that, or click on username under PLACES then click on enosServer folder.
Under DEVICES the enosServer Icon also has a unmount icon at the right side. DO NOT try to un-mount the enosServer by clicking on this.

sshfs uses the FUSE kernel module to fool your client computer into thinking the /server directory on enosServer is a local directory named /home/$USER/enosServer. Anything you can do on a local directory, you can now do on the remote computer's directory. This is how you will manage the content on the enosServer's DATA drive. Add files, delete files, add/delete folders just like you do on your local Drive. So, manage data files with sshfs, and do maintenance on the Operating System by using ssh pshare@enosServer. Once set up maintenance is usually just doing updates with pacman -Syu and data backups.

Use the following to unmount enosServer. In the Client computer type in the following:
$ fusermount -u /home/$USER/enosServer OR $ fusermount -u enosServer
Direct Thunar to the /home/$USER/enosServer directory, and it should be empty. Because there isn't a remote directory mounted there anymore.

AUTOMATE THE MOUNTING PROCESS

This is all fine and dandy, but who wants to type in these commands all the time, much less remember the exact Syntax for the commands? Let's automate this process some. Here are the instructions for XFCE, it should be doable in any Desktop Environment.

Mount enosServer with a desktop launcher

IN THE CLIENT COMPUTER as user in user's home directory

```
  $ ls -l
  drwx------. 2 username username 4096 Jan 31 20:34 bin
```

Look for a directory named bin. If bin doesn't appear, use mkdir to create one

```
$ mkdir bin
```

After creating bin, you may want to use

```
  $ chmod 700 bin
```

to change the permissions as above for security reasons.

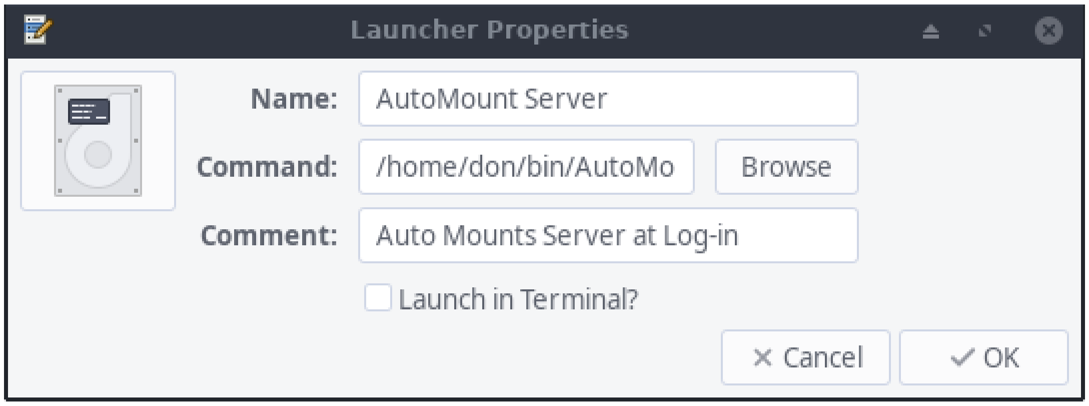
```
 $ cd bin
```

Using your favorite text editor, create a file named AutoMountServer.sh and add these 3 lines

```
  $ vi AutoMountServer.sh
     #!/bin/bash       (the first two characters are # then an exclamation mark)
     sshfs pshare@enosServer:/server   /home/$USER/enosServer
     exit              (type in exit as the last line of the file, then close file)
```

```
 $ chmod 754 AutoMountServer.sh     (make AutoMountServer executable)
 $ ll
   -rwxr-xr- -   don don  47 Jan 8   AutoMountServer.sh
```

```
$ sudo pacman -S alacarte
```
click on EndeavourOS at the left of the panel – Settings – Main Menu
In the left hand column, select System – then click on + New Item



Name: AutoMount Server
Command: /home/don/bin/AutoMo
Comment: Auto Mounts Server at Log-in

Fill in Name: with what ever you want to call this launcher
Fill in Command: Browse to /home/username/bin/ AutoMountServer.sh
Fill in Comment: with whatever you want
Launch in Terminal? Leave it unchecked
Click on image at the far left and navigate to a desired icon
+ Other Locations – Computer – usr – share – icons – Adwatia – 32x32 – devices – drive-harddisk.png
select the icon, back in Launcher Properties click OK

Log off, then Log in so XFCE4 can detect the changes.
home/enosServer should be empty.
Click on EndeavourOS button – System – AutoMountServer
The server should be mounted on enosServer

You could right click on the launcher and add it to the Desktop or the Panel.

Automatically Mount Server at Log in

It would be nice to have AutoMountServer run every time you log in.

Go to "EndeavourOS" in the panel, click on "Settings", click on "Session and Startup"
Select "Application Autostart" and hit the +Add.
Name: MountServer
Description: Auto Mount enosServer
Command: Navigate to /home/$USER/bin/AutoMountServer.sh
Trigger: on login

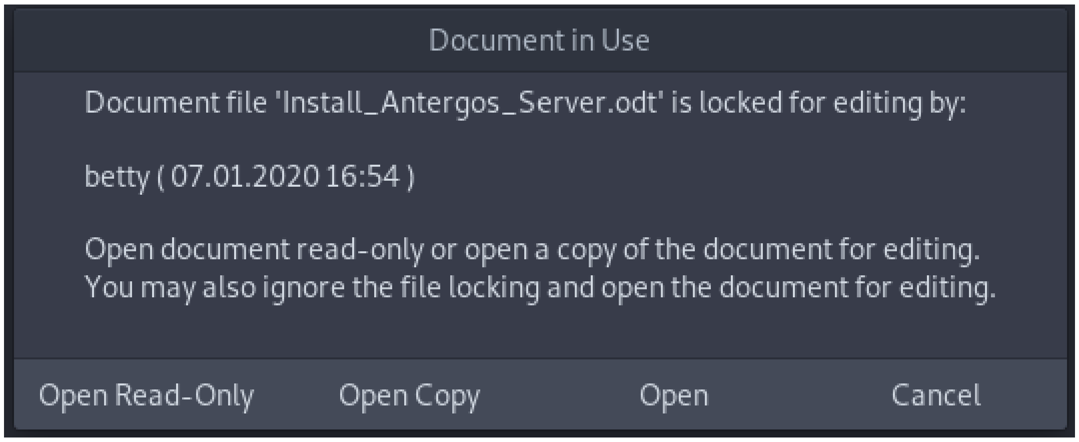
Reboot, and enosServer should be automatically mounted at login. Extremely simple for any level of Linux user. Once set up for a user, all that user has to do is log in and enosServer is ready to use. Check that Thunar and the enosServer are mounted.

This is how I set up my server access for my convenience, otherwise the wife would be continuously asking “How do you do that again?”.

The variations are many. If you don’t want the server mounted without entering a password, then don’t create any SSH keys. Now when the server is auto-mounted or mounted by clicking on the launcher, you will be prompted for pshare’s password.

If you don’t want the server to auto-mount at login, don’t add it to Session and Startup. With no SSH Keys and no auto-mount at log in, the server would not mount until you manually launched it and then entered the password.

At least in LibreOffice, if someone else has a file open when you try to open it you get:


Document in Use

Document file 'Install_Antergos_Server.odt' is locked for editing by:

betty (07.01.2020 16:54)

Open document read-only or open a copy of the document for editing.
You may also ignore the file locking and open the document for editing.

This is a good thing if multiple people in the household share files. If this happens, I would suggest clicking Cancel.

Try to avoid shutting down the computer with a file open on the server. Next time you try to open that file, you may get a similar warning as above saying the file is locked for editing by you. Improper shut downs can leave a lot of things hanging.

Configure any other Linux machines you have exactly like we did this one. Yes, all Linux Client machines will mount the server as user pshare. You could create different users on the server that match the user names of all users accessing the server. In fact, that is what you would want to do in an Enterprise environment.   But that can make administration a nightmare, plus this can cause a lot of permission problems between users. We want to keep our home file server as KISS as possible. This way every file and every directory on the server will have pshare as the owner and pshare as group. When we set up the Samba server, we will mount SMB shares as user pshare. That way when someone writes files from a Windows machine, the files will still have pshare as the owner and pshare as group. Everything will be very homogeneous on the Data Drive. Out of the box, SSH will allow up to ten simultaneous connections per SSH server IP address, in this case 192.168.0.1.150 most households don't have ten Linux boxes. But if you need more, edit the sshd_config file on the server and change MaxSessions to something higher than ten. I don't know how many simultaneous SSH sessions a single user (pshare) can have, but I have never exceeded it so far. As an experiment, I had 3 Windows computers and 4 Linux computers simultaneously on the server, all 7 streaming music, and performing other server tasks. The server never missed a beat. For a simple home server, it works great.

Manually Installing a USB 3 SSD for the DATA Partition

To install a SSD in a USB 3 enclosure as an external DATA partition. This SSD can be any size you want. Then this USB 3 SSD can be mounted during boot up in /etc/fstab as /server.

Power off the SBC ARM server computer and connect a USB 3 external enclosure with a SSD installed. Boot up the computer and login as root.

```
# cd                                ( go to /root directory )
# lb        (Odroid N2 should look similar to this, Odroid XU4 will have one mmcblk partition)
NAME           FSTYPE     FSSIZE     LABEL     MOUNTPOINT
sda
└─sda1             ext4                        XXXXX
mmcblk0
├───mmcblk1p1    vfat          252M       BOOT       /boot
└───mmcblk1p2    ext4          56.8G      ROOT       /
```

/dev/mmcblk1 is our OS device. /dev/sda is our target device. If the device is brand new and has never been partitioned it may look different. We have determined that /dev/sda is our new device (your setup may be different such as /dev/sdb etc)

Partition the USB SSD as one partition that uses all the space in the SSD
CAUTION: This WILL erase ALL DATA on your SSD.

```
# fdisk /dev/sda                   ( or adjust to /dev/sdb or whatever is relevant )
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 ext4 signature.        (this warning may not appear, if so yes)
   do you want to remove the signature?  yes
Command: w                                        (write partition table to disk and exit)

# mkfs.ext4 -L DATA /dev/sda1       (format our new partition to ext4, -L DATA is the label)
```

Modify /etc/fstab

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

```
# lsblk -o NAME,FSTYPE,LABEL,UUID  /dev/sda
NAME      FSTYPE    LABEL     UUID
sda
└─sda1     ext4          DATA      373fec37-5b49-44ba-b618-0009613e3ca0
```

you should see /dev/sda1 with a nice label of "DATA" and its UUID number.
Copy the UUID number on a sheet of paper.

```
# cp /etc/fstab /etc/fstab.orig          (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,noatime  0 2
```

close /etc/fstab

We need to create a mount point for the external DATA SSD and the external BKUP SSD

```
    # mkdir /server /serverbkup
```

The /server directory is used for mounting the USB SSD DATA partition The /serverbkup directory is used for mounting the USB BACKUP SSD partition. You should never put any files or sub-directories in these reserved directories.

```
# mount -a       (should show no errors if fstab is correct, if not edit /etc/fstab again)

# df -h /dev/sda1                              (should show our new /server )
   Filesystem  Size     Used    Available     Use%    Mounted on
   /dev/sda1    229G    61M       217G          1%       /server
```

If sda1 now shows a mount point of /server then the fstab is configured correctly.
If not, try editing /etc/fstab again.

IMPORTANT: Set ownership and permissions on our mounted external SSD device

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

You should now have something similar to this snippet.

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

Ensure the directories have the correct ownership (root users) and permissions drwxrwxr- -
If they are not correct, repeat the above commands until they are correct.

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

```
# systemctl reboot
```

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, and it will take a while, try to get a Console window   Ctrl-Alt-F2

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.

```
# ll  /server
   drwx------ 2 root root 4096 Jun 24 11:51  lost+found
                              (directory lost+found was created with mkfs.ext4 – we don’t need it)
#  rm -rf  /server/lost+found
# su pshare                    (Switch User to pshare, see NOTE at bottom of page)
$ echo “This is a test” > /server/test
$ ll /server
   -rw-r—r--  1  pshare pshare  15 Jun 24 14:02  test
$ cat /server/test
   This is a test
$ exit                ( then exit twice to log out of enosServer back to the Console window tty)
# exit
```

NOTE: On it’s own separate SSD you have a working partition at /server for all your data. Always work in /server as a user. Everything in /server should belong to user pshare, including all files and all directories.

Flash an eMMC card on Odroid XU4

On an Odroid XU4, if you opt to use a emmc card as the boot device, then additional steps are necessary. There are many opinions on micro SD VS emmc. The choice is yours.

To create an emmc card, first create a micro SD card as above including booting up the Odroid-XU4, installing keys, and update. Now that everything is working up to this point, place the emmc card on the emmc to microSD converter card. Then into the USB SD READER. Boot up the x86_64 computer with the latest EndeavourOS ISO.
Now repeat the above steps for the micro SD card but with the emmc card. When you shut down the x86_64 computer come back here and do the following.
Set the boot switch on the Odroid-XU4 board next to the HDMI jack to the uSD position (to the left). Ensure the micro SD card is in the XU4 SD slot. On the back side of the Odroid-XU4, Connect the eMMC module to the XU4, ensuring you hear a click when doing so.
Connect a monitor, keyboard, ethernet, & apply 5VDC.
The default user is *alarm* with the password *alarm,* the default root password is *root.*
Login as root

```
# cd /boot
# sh sd_fusing.sh /dev/mmcblk0
```

fusing should take place without errors. Poweroff the Odroid. Remove the micro SD card. Switch the boot selector switch to emmc (to the right). Power up the Odroid. You should now be running on the emmc card. Login as root.

```
    # pacman-key --init
    # pacman-key --populate archlinuxarm
    # pacman -Syu
```

Now go back to page 4 “Install EndeavourOS on ARM” section and continue installation.