INSTALL ARCH LINUX ARM FOR A SIMPLE LAN SERVER

The Hardware

At the top, two possibilities for the DATA drive which will store all your files. A USB-3 SSD external housing with a SSD of your choice and size. In this case a Samsung 850 500GB SSD. OR if more storage is needed, a USB-3 3.5 inch external hard drive enclosure, in this case with a 2 TB WD Red Label hard drive. Next is a 5 VDC 6 Amp power brick for the Odroid-XU4, not shown is a 12 VDC power supply included with the 3.5 inch hard drive if used. At the bottom, a USB SD Reader. A micro SD to SD adapter. A micro SD card OR an emmc to micro SD adapter with an emmc card for the Operating System. Then the Odroid-XU4 SBC (Single Board Computer). With optional Real Time Clock battery, 40 mm cooling fan, and a 1/4 inch plywood base with stand offs to protect the components on the bottom.

Of course, any ARM device that Arch Linux Arm supports can be used, such as a Raspberry PI 4 or the new Odroid N2 with 4 GB RAM, or what ever. 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.

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.

The storage device for the Operating System will be either a Micro SD card or an emmc card. The storage device for the DATA device will be by necessity a SSD or Hard Drive in an USB 3 enclosure as pictured above. A third storage device will be necessary for doing backups of the DATA drive. This backup device will also need to be in a USB 3 enclosure. This device would ideally be the same as the DATA device as far as manufacturer, model, and especially the size in Gigabytes or Terabytes.

Flash the micro SD card with the OS and bootloader.

The micro SD card must be at least 16 GB, 32 GB gives more head room for logs and etc, and anything over 32 GB is mostly wasted. 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 I have

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.

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. The ISO is not persistent, so after a reboot, it's all gone.

Boot into the msdos/MBR version of the EndeavourOS installer ISO.

Insert the USB SD READER containing the Micro SD to SD adapter with the micro SD card. In the Endeavour welcome screen, select Partition Manager (Gparted)

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/sda. Write this down.
Click on "Device" tab, and create a msdos Partition Table.
Click on "Partition" tab, then new
  Free Space preceding MiB: 2 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
Apply All Operations
Close GParted
IMPORTANT: Free Space preceding (MiB): 2 (MUST change from 1 to 2)

Close the Welcome screen
Open a terminal window.
```
$ 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/sda1)
# wget http://os.archlinuxarm.org/os/ArchLinuxARM-odroid-xu3-latest.tar.gz
# bsdtar -xpf ArchLinuxARM-odroid-xu3-latest.tar.gz -C MP
# cd  MP/boot
# sh sd_fusing.sh /dev/sdX      (change /dev/sdX to what's appropriate. e.g. /dev/sda)
# cd ../..
# umount MP
# exit
$ exit
```

The Arch Linux ARM Operating System is now installed.
NOTE: To install OS on an emmc card, see "Flash an emmc card" at the end of this tutorial.

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, & apply 5VDC

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 Odroid boots up:
Login as root (enter root for the username and root for the password)
```
# pacman-key --init
# pacman-key --populate archlinuxarm
# pacman -Syu
# reboot
```

Configure the Operating System for a Server

Login to the Odroid-XU4 as root.

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

Set up a user named pshare (for Public Share) and change root password

```
  # passwd                  (change the default root password to something more secure)
  # userdel -r alarm                  (delete the default user from the image)
  # useradd -m -s /bin/bash -u 1000 pshare      (add user pshare)
  # passwd pshare                     (set the password for user pshare)
  # gpasswd -a pshare users           (add pshare to group users IMPORTANT)
  # id pshare
    uid=1000 gid=1000 groups=1000(pshare),985(users)
  # groups pshare
    users pshare
```

Add the alias ll for ls -l command

I prefer the alias ll (List Long) installed as an alias for ls -l 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 of the OS by typing in “exit” and log back in to activate the new alias. Enter ll and it should complete without any error. If you get "command not found" you made a typing error while entering the alias. Try again.

Change hostname

```
  # hostnamectl set-hostname enosServer.localdomain  - -static
  # hostnamectl status
    Static hostname:  enosServer.localdomain
```

Setup a Static IP Address for the server

Servers work much better with a static IP adress than using a dynamic IP adress with DHCP. Set a static IP address for this server. Run “ip addr” to list current DHCP ip address and the ethernet interface name.

```
# ip addr
```

Out of that mess, look for the items highlighted red in these two lines

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> YaDa YaDa
inet 192.168.0.105/24 brd 192.168.0.55 scope global dynamic eth0
```

eth0 is the ethernet interface name, 192.168.0.105 is the IP address, /24 is the netmask, and dynamic indicates the IP address was assigned by DHCP and is dynamic.

```
# ll /etc/systemd/network
```

Find the config file to edit. It should be Ethernet-Interface-Name.network)

```
eth0.network                    (It may vary in different cases)
```

Edit the eth0.network file. Comment out the "DHCP=yes" line & make the following changes

```
# vi or nano /etc/systemd/network/eth0.network     (whatever your network config file is)
    [Match]
    Name=eth0

    [Network]
    # DHCP=yes                          (comment out with a leading #)
    Address=192.168.0.150/24            (your desired static IP address plus broadcast)
    Gateway=192.168.0.1                 (the IP address of the router this is connected to)
    DNS=192.168.0.1        ( Most routers have DNS pass through use same IP as Gateway)
    DNSSEC=8.8.8.8                    (secondary DNS 8.8.8.8 = DNS server at Google)
# systemctl reboot
```

After reboot, check the changes to # id pshare, # hostnamectl status, # ip addr

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> YaDa YaDa
   inet 192.168.0.150/24 brd 192.168.0.55 scope global eth0
```

Notice that the IP address is your static IP you entered, and dynamic no longer appears.

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, 1.3 GB.

```
# pacman -Q > enosPkgs
# wc -l enosPkgs
  126 enosPkgs            ( 126 is the number of installed packages)
# more enosPkgs           (to look at the package list)
```

SETUP THE SSH SERVER

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 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 the ports not listed 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

```
# 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 connect a USB 3 external enclosure with a SSD or 3.5 inch Hard Drive installed. Boot up the server computer. Login as root

```
# blkid
/dev/mmcblk1p1: UUID="d026ab30-1a28-4e18-8bca-6b07b05a03c9" TYPE="ext4"
/dev/sda1: UUID="2dbbf1ae-d7b8-4209-8265-89fcccc6cdac" TYPE="ext4"
```

/dev/mmcblk1p1 is the single partition for our OS device. The newly added device is /dev/sda1 If the device is brand new and has never been partitioned it may look different. Now that it is determined that /dev/sda is our DATA device

```
# fdisk /dev/sda
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 table to disk and exit)
```

```
# mkfs.ext4 -L DATA /dev/sda1
```

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

```
   # cd /                        (Change to root directory)
   # mkdir /server
```

You should now have something similar to this snippet.

```
# ll
drwxr-xr-x  46 root root 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 for the DATA SSD partition.
```
# blkid
/dev/mmcblk1p1: UUID="d026ab30-1a28-4e18-8bca-6b07b05a03c9" TYPE="ext4"
/dev/sda1: LABEL="DATA" UUID="b4dc7162-fcde-4b28-b8b9-e98626932902" TYPE="ext4"
```

you should see /dev/sda1 with a nice label of "DATA" and its UUID number
write down the UUID number without the quotes

Using vi or nano, add the following line at the end of the /etc/fstab file
The /etc/fstab file will probably not have any entries, just a couple of informational lines.

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

Reboot the computer. If boot up is normal, 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.

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

reboot
```
 # ll /
```

check the permissions (drwxrwxr--) and ownership (root users) and make sure they are correct. If they are correct, then fstab is doing it's job.
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.
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 fomatting, at least on 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 enough)
  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 and keyboard can now be removed to run the server headless. Maintenance can be performed in a Linux Client using SSH.

FYI, as if 01/07/2020 the latest ARMv7h kernel is 4.14.157-1-ARCH. Which is good for a server as the latest and greatest isn't always the best thing for reliability.

To make this tutorial 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 ARM repositories. No third party software. That is KISS to the utmost.

The following is purely informational and is not necessary to read. If you are curious what was just done, read on.

The Odroid-XU4 can be booted from either a micro SD card, or from an emmc card. The bootloader does not use Grub or other such thing to let you choose what you want booted. Instead, it looks at a hardware switch to determine whether to boot from the SD or emmc card.

The first step was to format the micro SD card with GParted. The SD was given a msdos Partition Table. Then one partition was created that started at 4096 sectors in, which is 2 MiB. Then the single ext4 partition started at 2 MiB in and used all the remaining sectors. The ext4 partition is for the OS, and the hidden first 2 MiB is for the boot loader, and other essential housekeeping stuff.

In a terminal window, a folder named Odroid was created and we switched to that folder. This just isolates what we are doing from the rest of the ISO filesystem. Once in that folder, we switched to the root user. As root, a mount point was created to mount the micro SD card named tmpMP for temporary Mount Point. Then the micro SD was mounted. Wget downloaded the latest Arch Linux Arm image for the Odroid-XU3. The Odroid-XU3 and the Odroid-XU4 use the same CPU, but they only support one image. Then the image was untarred and flashed to the micro SD card partition mounted at tmpMP. Now the operating system is in place on the single ext4 partition of the micro SD card. We still need a bootloader and other goodies to run the OS.

We cd into the mounted micro SD card's /boot directory. In that directory, the image put a script name sd_fusing.sh This script installs the bootloader and other stuff on the reserved first 2 MiB of the SD card. The process in now complete. We did a cd out of the mounted SD so the SD card is not "busy". Then the SD card was unmounted. If your plans are to use a SD card for the OS, you can install the SD on the Odroid, set the boot selector switch, and boot up.

Flash an emmc card

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

```
# sh /boot/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 The "Configure the Operating System for a Server" section and continue configuration.