Install a base EndeavourOS for a Simple Home Server

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.

Other information needed

Install current kernel or LTS kernel
Desired hostname: the instructions use enosServer, change it if you want
Desired user name: the instructions use pshare, change it if you want
Desired Last Triad of Static IP address for this computer:
Desired SSH port number
root password
user password

On a x86_64 computer that will become the server, hook up a monitor, keyboard, and mouse. After installation, the monitor, keyboard, and mouse can be disconnected and the server can be run headless. Connect a small SSD or HD on SATA 1 Anything over 32 GB will work fine. The smallest SSD they sell lately is 120 or 128 GB which can be obtained for about $25 USD. Of course anything laying around in the parts bin will work.

Boot up the latest EndeavourOS install ISO. During Bootup, press Delete key or F2, and disable the WiFi in the firmware if possible. Choose whether to boot the msdos/MBR version or the UEFI version of the installer. For a LAN server, I recommend msdos/MBR, but you can choose which ever you desire.

During testing, a 32 GB USB thumb drive was used for the OS, and another 32 GB USB thumb drive for the DATA drive. Not practical, but OK for testing before obtaining permanent hardware.

Open a terminal window as liveuser
$ git clone https://github.com/pudges-place/EndeavourOS-baseinstall.git
$ cd EndeavourOS-baseinstall

$ chmod 744 install* (make scripts executable)
$ ls -l
-rwxr--r-- root root install-base1.sh
-rwxr--r-- root root install-base2.sh

$ sudo ./install-base1.sh

Brings up the first page of the script, which has a description of what it does
and how to manually partition your mass storage device outside of the script.

If you want to partition your drive outside of this script, leave this page up as a guide while using Gparted or what ever. When partitioning is completed, continue with the script
OR
You can let the script automatically partition the selected storage device.
During formatting, "/dev/sda1 contains a ext4 file system, proceed anyway"
or something similar may occur, enter y and go on.

A notice will appear when finished
The user can re-enter arch-chroot and use pacman to add desired packages.
enter the sudo umount -a command, then
$ poweroff

POST INSTALL

When finished installing, remove install media and boot up then log-in as root.
See if networking is operational
ping -c 4 endeavouros.com (should get 0% packet loss)
ip addr (line 2: should show static IP you entered)
date
  Thu 23 Jan 2020 01:50:52 PM MST (check time and date are correct)
pacman -Q | grep keyring (to double check if endeavouros-keyring was installed)
pacman -Q | grep mirrorlist (to double check if endeavouros-mirrorlist was installed)

df -h the root directory will show approximately 2.2 Gbyte used
pacman -Q > pkglist
wc -l pkglist (will show approx 165 packages)
  165 pkglist
less pkglist (will display installed packages)

IMPORTANT

The install should have added pshare to the users group. As a check

```
# groups pshare
  users pshare
```

If users wasn't listed do the following to add pshare to users

```
# gpasswd -a pshare users
# groups pshare
  users pshare
```

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

```
# hostnamectl set-hostname MyServer.localdomain  - -static
# hostnamectl status          (to verify your new hostname if you changed it.)
```

SETUP A FIREWALL RULE to allow a Linux Client Computer access on
A rule must be set up to allow the Linux Client Computer access on SSH.

```
# ufw status                              (check status of ufw)
    status: active
```

Use the format: ufw allow from xxx.xxx.xxx.0/24 to any port XXXX
where xxx.xxx.xxx.0/24 is the Router's IP address block, XXXX is your chosen SSH port

```
# ufw allow from 192.168.0.0/24 to any port 9830
# 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. Anything coming in from an IP address outside of our private LAN is rejected. The entire world is blocked but any computer on our ethernet LAN is accepted on port XXXX.

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. Now is the time for the BIG SSD or Hard Drive. Boot up the server computer and log in 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 answer 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  774  /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)
# 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     /server
```

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

If it boots up normally, log back in 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- -  46 root users 4096 Aug 15 22:15 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 login and become root

```
# ll /
```

Recheck the permissions (drwxrwxr--) 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 enough)
 80.6M/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.

Since this is a script, the user can use a text editor to edit install-base1.sh and change the pacstrap entries to add or delete packages to be installed. Currently, this is lines 470 to 477. The script is meant to be the base for a EndeavourOS server, but by editing the pacstrap entries you could add DHCP, WiFI modules, or whatever and make it your own.

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-heasers, 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.

Three platforms used for testing

Left x86_64 ASRock motherboard with 8 GB RAM, 128 GB internal SSD for the OS installed in an mini ITX case with external USB SSD 250GB for DATA

Center x86_64 Atomic Pi motherboard with built in 2 GB RAM, 32 GB micro SD card for the OS with external USB SSD 250GB for DATA
This one is running and the Kill-A-Watt meter it's plugged into is reading 5 Watts at idle for both the motherboard and external SSD.

Right ARM Odroid-xu4 motherboard running Arch Linux Arm with built in 2 GB RAM, 32 GB micro SD card for the OS with an external USB WD Red Label hard drive at 2 Terabytes for DATA. The Odroid-XU4 with the Kill-A-Watt read 4 Watts and the USB Hard Drive has it's own 12 VDC power brick and the Hard Drive pulls 6 Watts. For a total of 10 Watts.

As mentioned earlier, a test was run with a 32 GB USB thumb drive for the OS and a 32 GB USB thumb drive for DATA. For a couple more possibilities view the first 10 min of https://www.youtube.com/watch?v=AtHzhtkxIc8
These are a little more difficult to install as there isn't any video connections it's all headless.