ARCH Linux Installation

This a guide to perform basic installation of an ARCH Linux distribution. Arch Linux is one of the most “Vanilla” distributions out of the many distributions available today.So the installation is not vey straight forward as other Linux distributions(Unless you know what you’re doing). This guide walks you through the installation of Arch Linux on a VMware workstation, but the installation steps should not differ on other virtualization platforms.

Index

1. Downloading the ISO

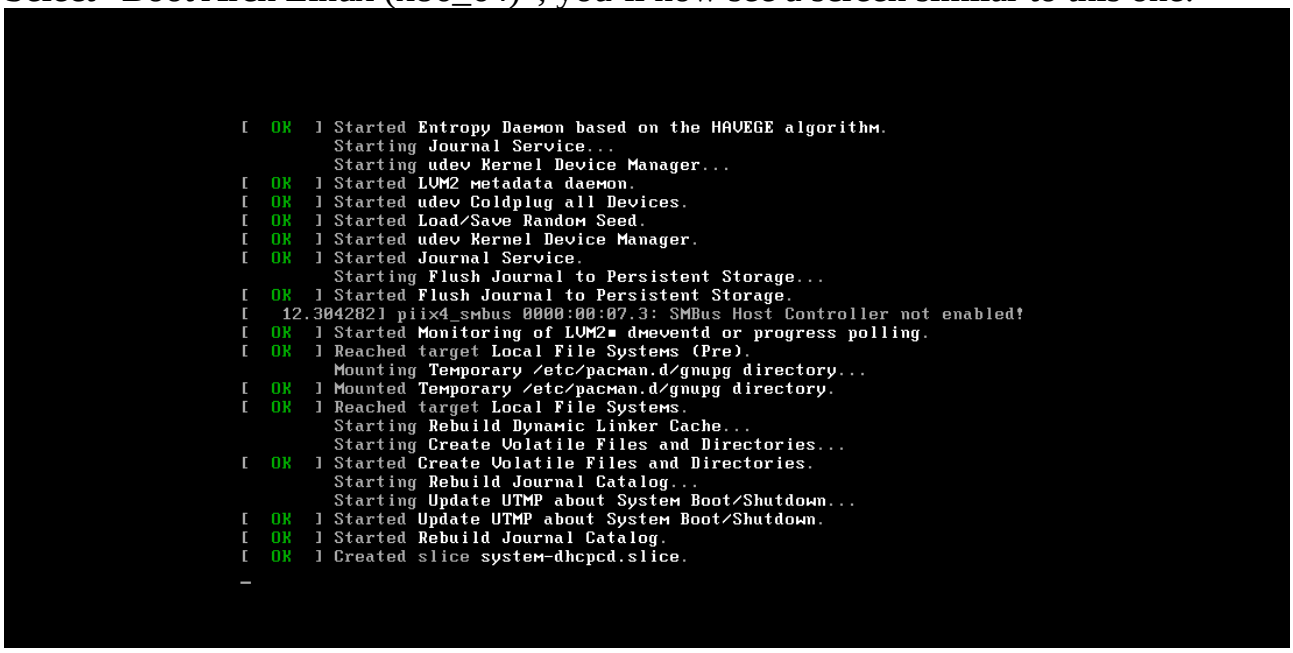
The ISO can be dowloaded from the official Arch Linux Website https://www.archlinux.org/download. It offers a direct download link and a magnet link to download it using a torrent client of your choice(typically transmission on Linux based desktops).

2.Booting from the ISO

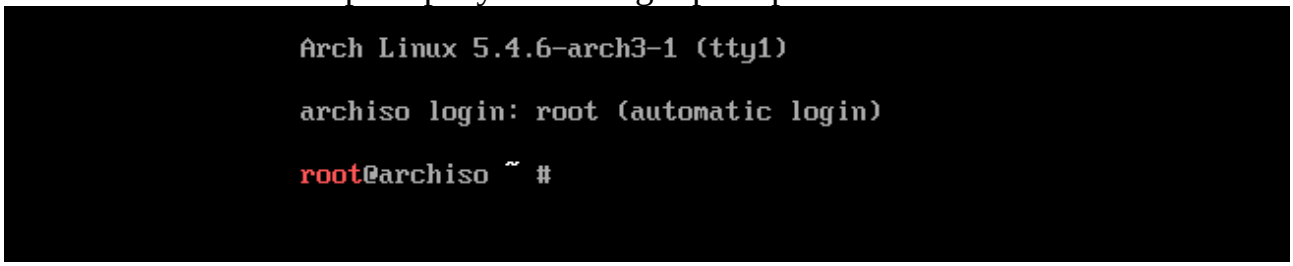
Before booting we need to create a VM in Vmware, I have used the below settings:
1 HDD 8 GB
1 GB RAM – 64 bit Operating Systems require atleast 512M of RAM.
2 Nics – One for connecting to the VM over ssh. The second which connects the VM to the internet.This is required because the packages are downloaded from the repositories available on the internet. The IP are assigned by DHCP.
1 CPU.

Once the VM is created similar to the above settings, attach the Arch linux ISO and boot it. You’ll see a screen similar to this one:

Select “Boot Arch Linux (x86_64)”, you’ll now see a screen similar to this one:



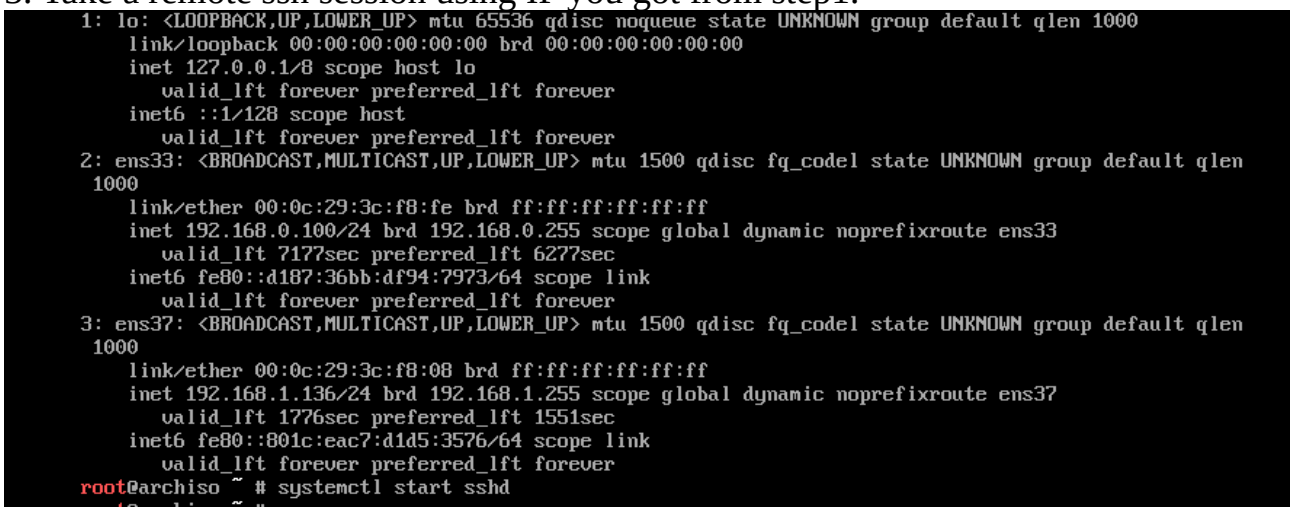
Wait until it boots and prompts you to a login prompt



3. Pre-Installation steps.

Before we start the actual installation we need to perform some pre-installation steps, so that we can take an ssh connection for the installation instead of doing it over the console.

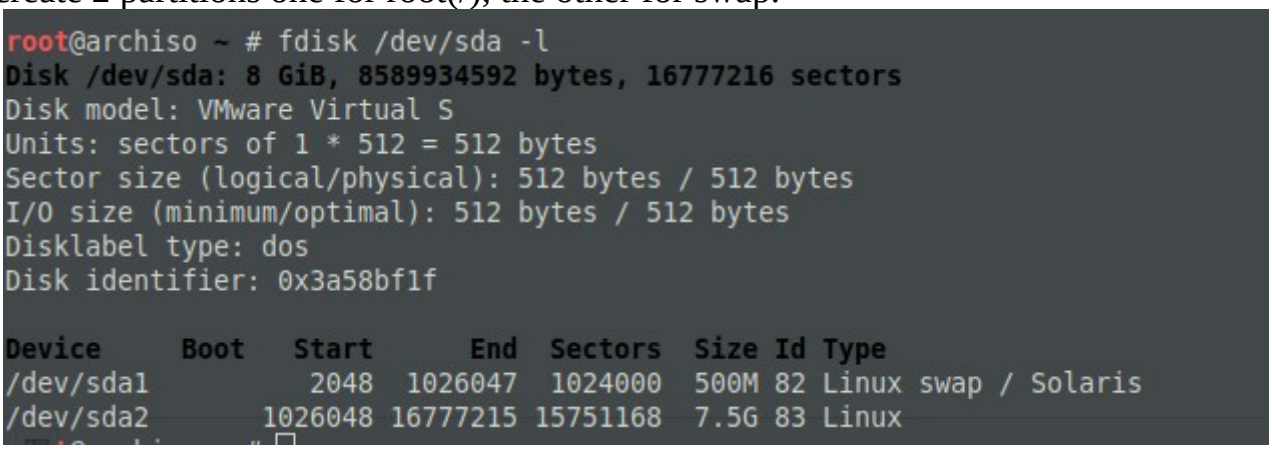
1. Check if the addresses are assigned.
2. Start the sshd service.
3. Take a remote ssh session using IP you got from step1.



4. Installation.

Now, that we've take care of most of the pre-requisites, without further ado, let's start the installation.

4.1. check the HDD added using the "fdisk -l" command. Using "fdisk" command create 2 partitions one for root(/), the other for swap.



4.2. Format the “Linux” partition with ext4. And the swap partition using the “mkswap” partition.

```
Device     Boot   Start      End  Sectors  Size Id Type
/dev/sda1          2048  1026047  1024000  500M 82 Linux swap / Solaris
/dev/sda2       1026048 16777215 15751168  7.5G 83 Linux
root@archiso ~ # mkfs.ext4 /dev/sda2
mke2fs 1.45.4 (23-Sep-2019)
Creating filesystem with 1968896 4k blocks and 492880 inodes
Filesystem UUID: 99c97f69-3521-4313-9e76-da8ba033895a
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

root@archiso ~ # mkswap /dev/sda1
Setting up swapspace version 1, size = 500 MiB (524283904 bytes)
no label, UUID=14af00dd-8f3f-4988-9f58-0590f5fe0d4d
root@archiso ~ # swapon -v /dev/sda1
swapon: /dev/sda1: found signature [pagesize=4096, signature=swap]
swapon: /dev/sda1: pagesize=4096, swapsize=524288000, devsize=524288000
```

4.3. Mount the patition (/dev/sda2) on a directory (/mnt).

```
root@archiso ~ # mount /dev/sda2 /mnt
```

```
tmpfs           90M     0   90M   0% /run/user/0
/dev/sda2       7.4G   34M  7.0G   1% /mnt
```

4.4 Start the installation of packages using the pacstrap command.

```
root@archiso ~ # pacstrap /mnt base base-devel grub vim bash-completion linux linux-firmware
==> Creating install root at /mnt
==> Installing packages to /mnt
:: Synchronizing package databases...
 core                                    134.5 KiB  62.9 KiB/s 00:02 [##########################################################] 100%
 extra                                    80.0 KiB  91.0 KiB/s 00:17 [##----------------------------------------------------------]   4%
```

Earlier, the “base” respository used to install the kernel and related packages, but now it does not hence it’s necessary to install the packages separately as shown above.

The completion of the installation of packages will show a message as below:

```
perl: warning: Falling back to the standard locale ("C").
(13/14) Updating the info directory file...
(14/14) Rebuilding certificate stores...
pacstrap /mnt base base-devel grub vim bash-completion linux linux-firmware  53.82s user 26.53s system 2% cpu 1:06:53.55 total
```

4.5. Populate the “/mnt/etc/fstab” file, as our root(/) is still /mnt. The “U” switch ensures the entries are populated using the UUID’s.

```
root@archiso ~ # genfstab -pU /mnt >> /mnt/etc/fstab
root@archiso ~ # cat /mnt/etc/fstab
# Static information about the filesystems.
# See fstab(5) for details.

# <file system> <dir> <type> <options> <dump> <pass>
# /dev/sda2
UUID=99c97f69-3521-4313-9e76-da8ba033895a       /               ext4            rw,relatime     0 1

# /dev/sda1
UUID=14af00dd-8f3f-4988-9f58-0590f5fe0d4d       none            swap            defaults        0 0
```

For the next steps we need to chroot to /mnt, so that it becomes our root. Use the below command.
arch-chroot /mnt

4.6 Install Grub and initrd.

```
[root@archiso ~]# grub-install /dev/sda
```

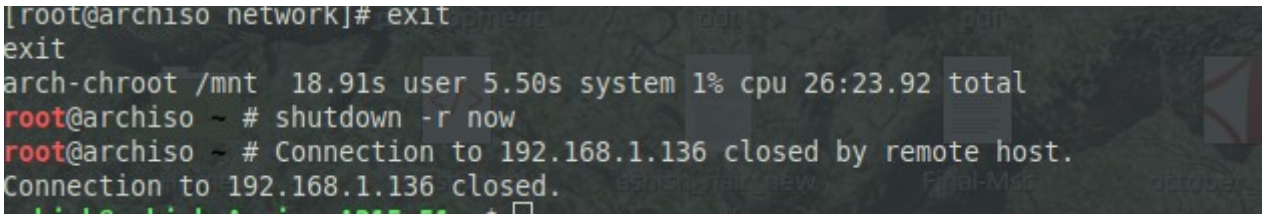
```
[root@archiso ~]# mkinitcpio -P
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'default'
  -> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux.img
==> Starting build: 5.4.14-arch1-1
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [autodetect]
  -> Running build hook: [modconf]
  -> Running build hook: [block]
  -> Running build hook: [filesystems]
  -> Running build hook: [keyboard]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-linux.img
==> Image generation successful
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'fallback'
  -> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux-fallback.img -S autodetect
==> Starting build: 5.4.14-arch1-1
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [modconf]
  -> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: aic94xx
==> WARNING: Possibly missing firmware for module: wd719x
  -> Running build hook: [filesystems]
  -> Running build hook: [keyboard]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-linux-fallback.img
==> Image generation successful
[root@archiso ~]#
```

Write the grub and initrd configuration to the “grub.cfg” file.

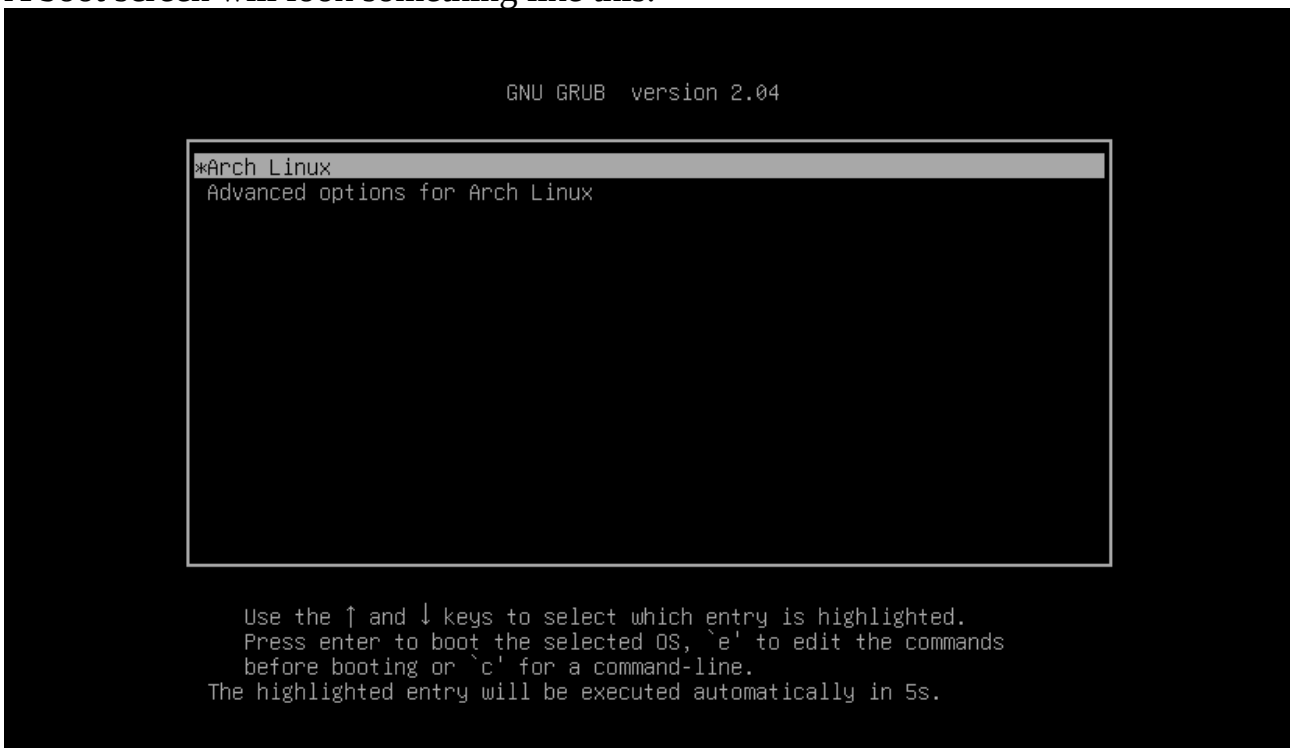
```
[root@archiso boot]# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-linux
Found initrd image: /boot/initramfs-linux.img
Found fallback initrd image(s) in /boot: initramfs-linux-fallback.img
done
```

We’re pretty much in a state where we can boot the system. Reboot the system and check if it’s booting. Next we’ll complete the post-installation steps (In the Post–Installation steps section) setting the hostname, Keyboard layout, keymap, locale

setting,Time-zones, persisting the network configuration files.For now, let’s exit from the chrooted environment and reboot it.

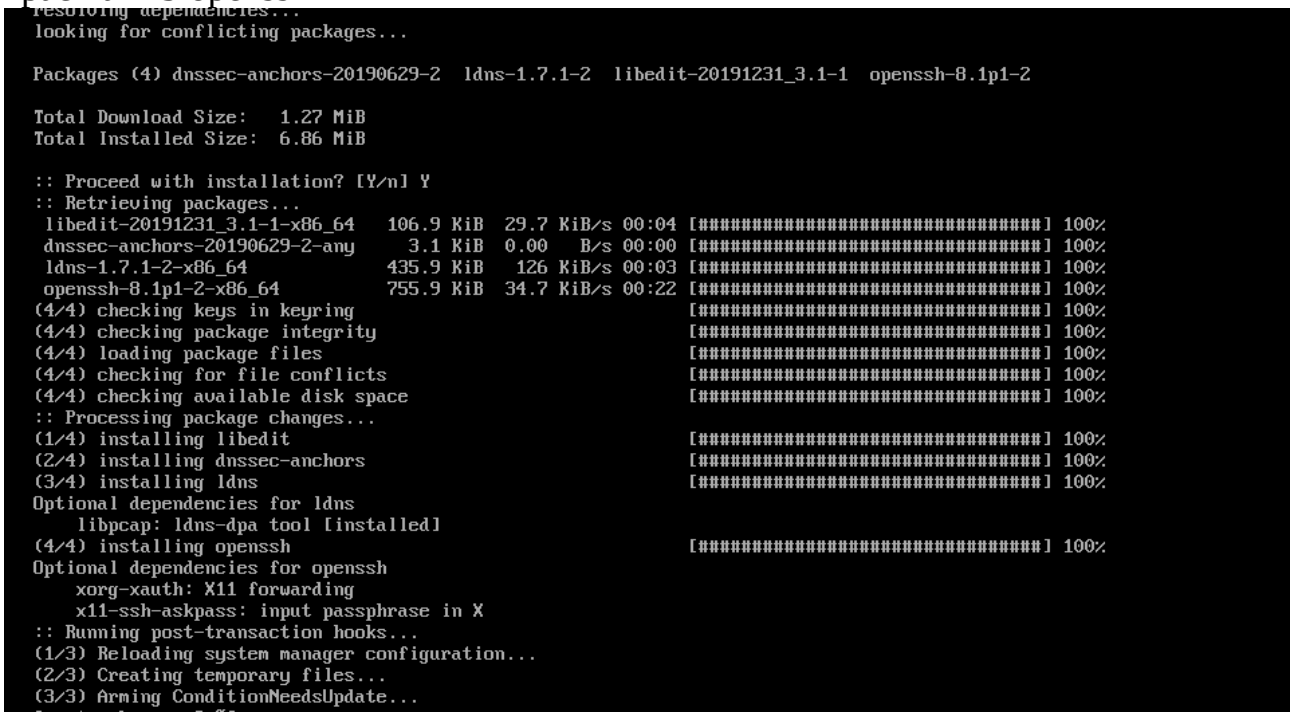


A boot screen will look something like this.

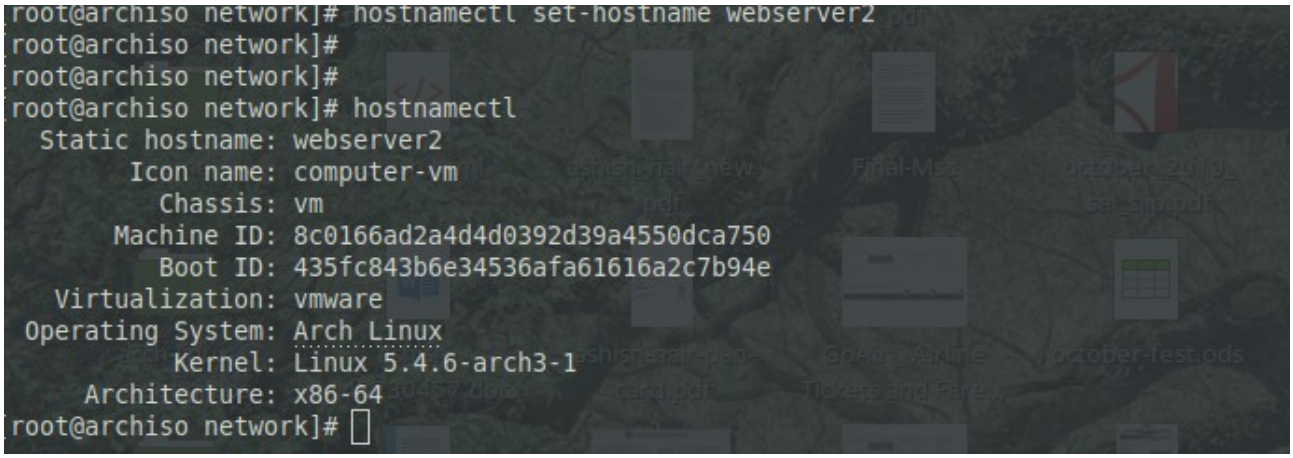


5 Post-Installation steps.

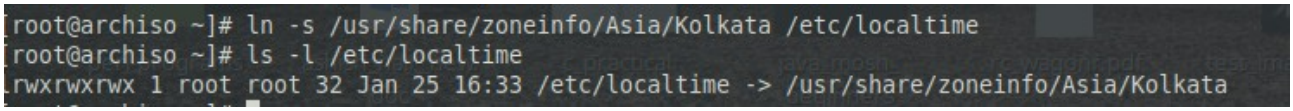
5.1 let's start by installing the ssh package, so that we can perform the rest of the steps over ssh, instead of doing it from the console. The command to install is "pacman -S openssh"



5.2 Next we'll set the hostname. I have used the hostnamectl command.



5.3 Set the timezone.



5.4 Generate the locale settings use the below procedure:
1. Edit /etc/locale.gen, uncomment your locale. Based on where you're located. For me it's LANG=en_US.UTF-8. Save and exit.
2. Then run the "locale-gen" command

3. set it using the localectl command : localectl set-locale LANG=en_US.UTF-8
4. set the keymap : localectl set-keymap us
5. check the settings using : localectl show

```
[root@archiso ~]# localectl status
   System Locale: LANG=en_US.UTF-8
       VC Keymap: us
      X11 Layout: us
       X11 Model: pc105+inet
     X11 Options: terminate:ctrl_alt_bksp
```

5.5 Check your IP and perist it using configuration files.

```
[root@archiso network]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:3c:f8:fe brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.100/24 brd 192.168.0.255 scope global dynamic noprefixroute ens33
       valid_lft 6733sec preferred_lft 5833sec
    inet6 fe80::d187:36bb:df94:7973/64 scope link
       valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:3c:f8:08 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.136/24 brd 192.168.1.255 scope global dynamic noprefixroute ens37
       valid_lft 1332sec preferred_lft 1107sec
    inet6 fe80::801c:eac7:d1d5:3576/64 scope link
       valid_lft forever preferred_lft forever
```

```
[root@archiso network]# cat ens33.network
[Match]
Name=ens33
[Network]
DHCP=yes
[root@archiso network]# cat ens37.network
[Match]
Name=ens37
[Network]
DHCP=yes
```

Enable the netword unit .
systemctl enable networkd-systemd

5.6 Take a reboot to verify if everything’s fine and settings we made persists accross reboots.

```
[root@webserver2 ~]# uname -a
Linux webserver2 5.4.14-arch1-1 #1 SMP PREEMPT Thu, 23 Jan 2020 10:07:05 +0000 x86_64 GNU/Linux
[root@webserver2 ~]#
```

We’re now done with the installation and minimum settings required for a server to be up and running.