Creating a PXE Server

April 24, 2017

1 Purpose

The purpose of this document is to give a baseline instruction set of how to build a PXE and storage server suitable for use with Panucci.

1.1 End Goals

- PXE Server to boot client machines from
- NFS server to serve client root and images
- Client image
- NAT passthrough for network traffic

1.2 Outline of Path

1. Install Ubuntu Server on server machine
2. Configure Network Interfaces
3. Configure DHCP
4. Configure TFTP
5. Configure `pxelinux`
6. Configure NFS
7. Create client base
8. Configure client base for network boot
9. Copy client base to server
10. Finish configuring `pxelinux`
11. Test and boot

2 Instructions

2.1 Install Ubuntu Server on machine

1. Download Ubuntu Server (or your Linux distribution of choice – non-Ubuntu distributions are currently not discussed in this document, but the general steps are the same)

   Ubuntu Server can be downloaded from
   `https://www.ubuntu.com/download/server`
2. Create install media (USB drive or CD/DVD) using the ISO
3. Boot the intended PXE server from the boot media
4. Complete the install by following the prompts

   It is recommended to install the OpenSSH Server package set during installation to enable remote configuration and administration.
5. After installation is completed, reboot the server and log in.

2.2 Configuring Network

1. Get a list of all interfaces in the unit, using the following command

   ```
   ip addr | grep -E "[0-9]+: " | grep -ohE ": [0-9a-z]+:" |\
   sed -e "s/[: ]//g" | grep -v "lo"
   ```

   You'll receive output like this:

   ```
   lo
   enp134s0f0
   enp134s0f1
   enp135s0f0
   enp135s0f1
   enp138s0f0
   enp138s0f1
   enp139s0f0
   enp139s0f1
   enp1s0f0
   enp1s0f1
   ```

   This is a list of all network interfaces in the machine.
2. Open up `/etc/network/interfaces` in a text editor
   `sudo nano /etc/network/interfaces`

2.2.1 Loopback

The loopback device is just a testing device; its entry should always read:

```
# The loopback network interface
auto lo
iface lo inet loopback
```

2.2.2 Primary Network Interface

For the purposes here, we'll call whichever interface we'll be using to access the outside network the 'primary' interface. To determine which interface is your primary interface, plug in the port you want to use for outside network access and run the following:

```
ifconfig
```

If you're using DHCP, the only two interfaces that have interfaces up are `lo` on 127.0.0.1 and our primary interface.

Setting automatic addresses (DHCP) The code block for DHCP is pretty short, fortunately (replace eth0 with your interface):

```
auto eth0
iface eth0 inet dhcp
```

Setting static IP If you are using a static IP assignment (replace eth0 with your primary interface):

```
# The primary network interface
auto eth0
iface eth0 inet static
        address 10.0.2.240
        netmask 255.255.255.0
        broadcast 10.0.2.255
        network 10.0.2.0
        gateway 10.0.2.1
```

`address` should be the static IP for the interface. `netmask` should be the octal-notation netmask. `broadcast` should be the local broadcast IP. `network` should be the associated network. `gateway` should the IP of the gateway on the network

2.2.3 Setting DHCP interfaces

For each interface that should allow a DHCP server (replace eth1 with the interface name):

```
auto eth1
iface eth inet static
        address 192.168.135.1
        netmask 255.255.255.0
        broadcast 192.168.135.255
        network 192.168.135.0
```

`address` should be the DHCP server IP for that interface. Make sure that no two interfaces have the same address.

`netmask` should be the netmask for the DHCP range

`broadcast` should be the broadcast IP for the DHCP range

`network` should be the DHCP range's network

Save the file after you've adding all interfaces

2.3 Configure DHCP server

Note: DHCP is capable of breaking a network. Be careful while doing anything using DHCP can cause problems.

1. Install isc-dhcp-server (`sudo apt-get install isc-dhcp-server`)
2. Open /etc/dhcp/dhcpd.conf (`sudo nano /etc/dhcp/dhcpd.conf`)
3. Leave the first uncommented line alone unless your environment requires it:

   ```
   ddns-update-style none;
   ```

4. While it's not strictly necessary, feel free to set `domain-name` and `domain-name-servers` for now:

   ```
   option domain-name yourcompany.com
   option domain-name-servers 8.8.8.8 8.8.4.4
   ```

In the example above, the DNS servers are set to use Google's public DNS, but if your company requires a custom DNS, substitute it there.

5. Set your DHCP lease times.

```
default-lease-time 600;
max-lease-time 1800;
```

   Lease time is expressed in seconds.

6. For each network you attached to an interface in `/etc/network/interfaces`, add the following to the end of the configuration file:

```
subnet 192.168.134.0 netmask 255.255.255.0 {
  range 192.168.134.2 192.168.134.8;
  option broadcast-address 192.168.134.255;
  option routers 192.168.134.1;
  option subnet-mask 255.255.255.0;
  filename "pxelinux.0";
}
```

   Line by line

   (a) `subnet 192.168.134.0 netmask 255.255.255.0 {`
   Set the first group to the network specified in `interfaces`, and the group after `netmask` to the netmask specified for that network.

   (b) `range 192.168.134.2 192.168.134.8;`
   Set the range of the DHCP pool. These addresses must be on the same subnet as the interface. It's recommend to set the interface to be xxx.xxx.xxx.1, and start the range at xxx.xxx.xxx.2.

   (c) `option broadcast-address 192.168.134.255;`
   Set the broadcast of the network, in line with the one set in `interfaces`.

   (d) `option routers 192.168.134.1;`
   Set the router option to the interface's IP address.

   (e) `option subnet-mask 255.255.255.0;`
   Set the netmask that DHCP devices will take from the network.

   (f) `filename "pxelinux.0";`
   Set the `filename` that will be loaded during PXE boot.

   (g) `}`
   Be sure to close the bracket!

7. Save the file (in nano, `^X`)
8. Enable the DHCP server

   ```
   sudo systemctl enable isc-dhcp-server
   ```

9. Restart networking to configure interfaces

   ```
   sudo /etc/init.d/networking restart
   ```

10. Start the DHCP server service

    ```
    sudo systemctl start isc-dhcp-server
    ```

2.4 Configure TFTP booting

1. Install the TFTP server

   ```
   sudo apt-get install tftpd-hpa
   ```

2. Open `/etc/default/tftpd-hpa`

   ```
   sudo nano /etc/default/tftpd-hpa
   ```

3. Tell the tftpd-hpa service to use `/tftpboot` as its root by adding the following line:

   ```
   OPTIONS="-l -s /tftpboot"
   ```

4. Save the file.
5. Create `/tftpboot`

   ```
   sudo mkdir -p /tftpboot/pxelinux.cfg
   ```

6. Install syslinux and pxelinux

   ```
   sudo apt-get install syslinux pxelinux
   ```

7. Copy pxelinux to /tftpboot

   ```
   sudo cp /usr/lib/PXELINUX/pxelinux.0 /tftpboot
   sudo mkdir -p /tftpboot/boot
   sudo cp -r /usr/lib/syslinux/modules/bios /tftpboot/boot/isolinux
   ```

8. Open the pxelinux configuration file

   ```
   sudo nano /tftpboot/pxelinux.cfg/default
   ```

9. Add the following to the (likely empty) file:

```
DEFAULT Linux

LABEL Linux
KERNEL vmlinuz-4.8.0-36-generic
APPEND root=/dev/nfs initrd=initrd.img-4.8.0-36-generic \
devfs=nomount nfsroot=10.0.2.240:/nfsroot ip=dhcp rw
```

Line by line

(a) `DEFAULT Linux`
This sets the default for pxelinux to load. Without this, it will not load correctly. Set this to the `LABEL` you set below.

(b) `LABEL Linux`
The name associated with the following entry.

(c) `KERNEL vmlinuz-4.8.0-36-generic`
This is the kernel that will be booting. This will be corrected later.

(d) `APPEND root=/dev/nfs initrd=initrd.img-4.8.0-36-generic\ devfs=nomount nfsroot=10.0.2.240:/nfsroot ip=dhcp rw`
This tells pxelinux what the root is (using NFS for the root), which initramfs to load (this will be corrected later), what the NFS root is, and sets it for read/write and tells it the IP address will be assigned by DHCP.

10. Save the file
11. Change permissions for `/tftpboot`
    `sudo chmod -R 777 /tftpboot`

2.5 Configure NFS

1. Install `nfs-kernel-server`
   `sudo apt-get install nfs-kernel-server`
2. Create `/nfsroot`
   `sudo mkdir /nfsroot`
3. Open `/etc/exports`
   `sudo nano /etc/exports`
4. Add the following line:
   `/nfsroot 192.168.0.0/16(rw,no_root_squash,async,insecure)`
   Explanation The network should be the overall larger network (often a /16 if you're using multiple networks).

5. Save the file.
6. Start the export

   `sudo exportfs -rv`

2.6 Set up NAT

1. Open `/etc/rc.local`

   `sudo nano /etc/rc.local`

2. Add the following lines to the end, just above `exit 0`

   ```
   /sbin/iptables -P FORWARD ACCEPT
   /sbin/iptables -A POSTROUTING -o eth0 -j MASQUERADE --table nat
   ```

   Change eth0 to your primary interface.

3. Save the file.
4. Make the changes active by running the following commands:

   ```
   sudo iptables -P FORWARD ACCEPT
   sudo iptables -A POSTROUTING -o eth0 -j MASQUERADE --table nat
   ```

2.7 Install On Client

The easiest way to get a working image is to install Linux on a client machine, configure to your liking, and then copy the files to your NFS mount.

1. Install according to your distribution's instructions.
2. On your client machine, install initramfs-tools

   `sudo apt-get install initramfs-tools`

3. Copy the client's vmlinuz to its home:

   `sudo cp /boot/vmlinuz-‘uname -r‘ ~`

4. Open the initramfs tools configuration:

   `sudo nano /etc/initramfs-tools-initramfs.conf`

5. Add or change the `BOOT` flag:

   `BOOT=nfs`

6. Add or change the `MODULES` flag:

   `MODULES=netboot`

7. Generate a new initramfs

   `sudo mkinitramfs -o ~/initrd.img-‘uname -r‘`

8. Mount the NFS share on the client machine

   `sudo mount -t nfs 192.168.134.1:/nfsroot /mnt`

   Replace the IP address above with the IP address on whatever interface the client machine is connected to.

9. Copy the client root to the NFS root

   `sudo cp -ax /. /mnt/.`

10. Power down the client (and remove the hard drive if possible)

11. Return to the server machine

12. Copy the vmlinuz and initramfs files to `/tftpboot`

    `sudo cp /nfsroot/home/user/initrd.img-* /tftpboot/`

    `sudo cp /nfsroot/home/user/vmlinuz-* /tftpboot/`

    In each of the above, replace `user` with the user's name from the client machine.

13. Edit /tftpboot/pxelinux.cfg/default to reflect the correct filenames for `vmlinuz` and `initrd.img`

14. Edit the PXE client's `/etc/fstab`

    `sudo nano /nfsroot/etc/fstab`

    ```
    none /tmp tmpfs defaults 0 0
    none /var/run tmpfs defaults 0 0
    none /var/lock tmpfs defaults 0 0
    none /var/tmp tmpfs defaults 0 0
    ```

    This set up mounting /tmp and /var/{`run, lock, tmp`} as tmpfs so that they don't get sent back to the NFS mount. This prevents issues with multiple clients.

15. Inside the NFS root, delete the folders we'll be mounting as tmpfs

    ```
    sudo rm -fr /nfsroot/tmp
    sudo rm -fr /nfsroot/var/run
    sudo rm -fr /nfsroot/var/lock
    sudo rm -fr /nfsroot/var/tmp
    ```

This is necessary so that the tmpfs mount doesn't fail. Note that in the pxelinux.cfg/default above, devfs has the `nomount` option, so that the device table isn't duplicated between client machines.

16. Change the client network interface settings

    `sudo nano /nfsroot/etc/network/interfaces`

    Change any reference to DHCP to manual. This prevents the client machine from hanging during shutdown/reboot (the client machine will hang when an interface goes down at shutdown without this change).

17. Set manual nameservers

    The change to manual interface control, instead of DHCP, leaves client without a way to automatically get DNS servers. If the client should be able to see to the wider internet, it is recommended to set DNS servers manually.

    `sudo nano /etc/resolvconf/resolv.conf.d/base`

    Add or edit the file to include the following lines:

    ```
    nameserver 8.8.8.8
    nameserver 8.8.4.4
    ```

    In the above example, the nameservers are set to Google's nameservers (8.8.8.8 and 8.8.4.4). These are publicly accessible DNS servers that are viable for any situation, barring a specific need to set a custom DNS.

2.8 Restart

Restart the client; it should PXE boot now. Congratulations! You've successfully set up network booting!

3 Behind the Scenes

PXE as a standard is fairly well-defined.

1. During boot, the Option ROM on the Network Interface Controller (NIC) attempts to obtain an IP address using DHCP. Failing this step results in the machine continuing on its normal boot process.
2. If the DHCP lease is successful, the NIC checks for a specified bootable file.
3. If the bootable file specification exists, the NIC attempts to download the file using Trivial File Transfer Protocol (TFTP).
4. If the file is successfully downloaded, the NIC attempts to run it.
5. The boot file runs, loading any additional files over TFTP.

   In this instance, the boot file loads a configuration, which tells it to load initrd.img and vmlinuz into memory.
6. The loaded files ‘bootstrap’ the machine into working, loading a full operating system, mounting network filesystems, etc.
7. The client machine loads into a fully-functional copy of the boot image.