

Setting up a Server for PXE Network Booting

If you're looking to perform a lot of system recovery, or system installation, then network booting with PXE is ideal. PXE allows you to boot up a system and have it automatically get an IP address via DHCP and start booting a kernel over the network.

PXE itself stands for "Pre-boot eXecution Environment", which describes how it works in the sense that the clients using it haven't booted in a traditional manner.

In order to use PXE you need to setup a boot-server which will allow client systems to :

- Request an IP address (via DHCP)
- Download a kernel (via TFTP)

With both of these services in place any system which supports PXE/network-booting (you might need to enable it in the BIOS) should be able to gain an IP address, fetch a kernel, and boot without an installed operating system.

(This is ideal for systems which can't be booted by a traditional approach; for example your new AMD-64 system which doesn't have a CD/DVD drive!)

Our Setup

For the purposes of this article we'll assume:

- We're working with a small network 192.168.1.0/24
- We'll allow all local machines to boot and get an IP address via DHCP from the range 196.168.1.70-192.168.1.100
- Our "boot-server" is the host "itchy" with IP address 192.168.1.50
- We will serve the same kernel to each host.

In our example we'll configure a PXE-server which will allow remote systems to run the Debian Etch installer, but nothing here is specific to that. PXE allows you to configure a system to boot from an arbitrary kernel (and matching ramdisk if you wish to use one). With the correct configuration you can even cause the clients to mount a remote file-system via NFS and have a diskless thin-client system.

TFTP Setup

TFTP is a very simple file-transfer protocol, which is very similar to FTP but which doesn't use any kind of authentication. If you're not already running a TFTP server you can install one by running:

```
root@itchy:~# apt-get install tftpd-hpa
```

Once installed you'll need to enable it by editing the file /etc/default/tftpd-hpa. You should change RUN_DAEMON to yes, leaving you with contents like these:

```
#Defaults for tftpd-hpa
RUN_DAEMON="yes"
OPTIONS="-l -s /var/lib/tftpboot"
```

Now create the root directory, if it is missing, and start the server:

```
root@itchy:~# mkdir -p /var/lib/tftpboot
```

Setting up a Server for PXE Network Booting

```
root@itchy:~# /etc/init.d/tftpd-hpa start
Starting HPA's tftpd: in.tftpd.
```

Once our systems have retrieved an IP address via DHCP they will request files from beneath the `/var/lib/tftboot` root directory. We'll come back to the contents of this directory shortly.

DHCP Setup

If you don't already have a DHCP server configured upon your LAN you'll need to install one. If you're using a small home router, or similar, to provide local DHCP services you must disable this first. Since we require the DHCP server to pass back some extra options to clients which the majority of routers won't allow).

Discussing a full DHCP installation is mostly beyond the scope of this introduction but the thing we're trying to do is fairly simple. The goal of the DHCP server in this setup is twofold:

- We obviously want to use it to allow clients to request and receive an IP address.
- We want to cause the DHCP "answer" to give some extra details to the clients which are requesting an address:
 - The address of the TFTP server.
 - The initial filename to request from the TFTP server.

The most common DHCP server is the `dhcp-server` package, and you can install this by running:

```
root@itchy:~# apt-get install dhcp3-server
```

Once installed the server is configured in the file `/etc/dhcp3/dhcpd.conf`, and there are a lot of available options described there. For our example we'll use the following configuration:

```
option domain-name-servers 62.31.64.39, 62.31.112.39;
default-lease-time 86400;
max-lease-time 604800;
authoritative;

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.70 192.168.1.100;
    filename "pxelinux.0";
    next-server 192.168.1.50;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.1.255;
    option routers 192.168.1.1;
}
```

Here we've configured the server to give out IP addresses from the range 192.168.1.70-100, set the default gateway to be 192.168.1.1 and use our ISP's nameservers.

We've also used `next-server` to point to the TFTP server we're using (which is the same host as our DHCP server, but doesn't need to be). We've chosen the default name of `pxelinux.0` as the name of the file for booting clients to request.

Setting up a Server for PXE Network Booting

Using dnsmasq Instead

Personally I use the dnsmasq package to provide DHCP services upon my LAN, since this is small and simple and provides other useful abilities, setting up PXE booting with dnsmasq just requires the addition of the following line to /etc/dnsmasq.conf:

```
dhcp-boot=pxelinux.0,itchy,192.168.1.50
```

(Again we've setup the filename along with the name and IP address of the TFTP server which is "itchy" / 192.168.1.50 in this example)

Restarting the service after this change is as simple as:

```
root@itchy:~# /etc/init.d/dnsmasq restart
Restarting DNS forwarder and DHCP server: dnsmasq.
```

PXE Configuration

Now that we've configured the TFTP and DHCP servers we need to go back and complete the configuration. By default when a client boots up it will use its own MAC address to specify which configuration file to read - however after trying several options it will fall back to requesting a default file.

We need to create that that file, which will contain the list of kernels which are available to boot, we'll firstly need to create a directory to hold it:

```
root@itchy:~# mkdir /var/lib/tftpboot/pxelinux.cfg
```

Now save the following as /var/lib/tftpboot/pxelinux.cfg/default:

```
DISPLAY boot.txt

DEFAULT etch_i386_install

LABEL etch_i386_install
    kernel debian/etch/i386/linux
    append vga=normal initrd=debian/etch/i386/initrd.gz --
LABEL etch_i386_linux
    kernel debian/etch/i386/linux
    append vga=normal initrd=debian/etch/i386/initrd.gz --

LABEL etch_i386_expert
    kernel debian/etch/i386/linux
    append priority=low vga=normal initrd=debian/etch/i386/initrd.gz -
-

LABEL etch_i386_rescue
    kernel debian/etch/i386/linux
    append vga=normal initrd=debian/etch/i386/initrd.gz
rescue/enable=true --

PROMPT 1
```

Setting up a Server for PXE Network Booting

```
TIMEOUT 0
```

This file instructs the client to display the contents of the file boot.txt so create that too:

```
- Boot Menu -  
=====
```

```
etch_i386_install  
etch_i386_linux  
etch_i386_expert  
etch_i386_rescue
```

The only remaining job is to download the official Etch installer kernel and associated files and save them in the directories specified in the default file we created:

```
root@itchy:~# cd /var/lib/tftpboot/  
root@itchy:~# wget  
http://ftp.uk.debian.org/debian/dists/etch/main/installer-  
i386/current/images/netboot/debian-installer/i386/pxelinux.0
```

```
root@itchy:~# mkdir -p /var/lib/tftpboot/debian/etch/i386  
root@itchy:~# cd /var/lib/tftpboot/debian/etch/i386  
root@itchy:~# wget  
http://ftp.uk.debian.org/debian/dists/etch/main/installer-  
i386/current/images/netboot/debian-installer/i386/linux  
root@itchy:~# wget  
http://ftp.uk.debian.org/debian/dists/etch/main/installer-  
i386/current/images/netboot/debian-installer/i386/initrd.gz
```

When these commands have been completed we'll have the following structure:

```
root@itchy:~# tree /var/lib/tftpboot/  
/var/lib/tftpboot/  
|-- boot.txt  
|-- debian  
|   |-- etch  
|       |-- i386  
|           |-- initrd.gz  
|           |-- linux  
|-- pxelinux.0  
|-- pxelinux.cfg  
|   |-- default
```

```
4 directories, 5 files
```

(We only used debian/etch here in case we want to offer other installers in the future. You can put everything in one directory if you wish, just update pxelinux.cfg/default to match.)

We should now be ready to test the setup.

Setting up a Server for PXE Network Booting

We've installed a pxelinux.0 file which will instruct booting clients to request pxelinux.cfg/default. This will then make a list of boot options available, which are displayed by the simple boot menu file we created.

The files which are used for booting are stored beneath the TFTP root directory and thus accessible to booting clients.

Sample Run

A sample remote boot looks like this:

```
PXE entry point found (we hope) at 9AE5:00D6
My IP address seems to be C0A80146 192.168.1.70
FTFTP prefix:
Trying to load: pxelinux.cfg/01-00-14-22-a1-53-85
Trying to load: pxelinux.cfg/C0A80146
Trying to load: pxelinux.cfg/C0A8014
Trying to load: pxelinux.cfg/C0A801
Trying to load: pxelinux.cfg/C0A80
Trying to load: pxelinux.cfg/C0A8
Trying to load: pxelinux.cfg/C0A
Trying to load: pxelinux.cfg/C0
Trying to load: pxelinux.cfg/C
Trying to load: pxelinux.cfg/default
- Boot Menu -
=====

etch_i386_install
etch_i386_linux
etch_i386_expert
etch_i386_rescue
```

As you can see the system here attempted to load several configuration files, based upon its MAC address (01-00-14-22-a1-53-85) initially then falling back to the octets in the IP address it was given by DHCP (192.167.1.70).

Finally it managed to load a working configuration using the last-chance default file we created. This in turn instructed it to show the boot menu we created.

From here on the system will boot into whichever kernel you specify. (We could configure the system to timeout here and just boot into a default option, but we didn't.)

From here on you should understand how PXE can be used to boot an arbitrary kernel and initial ramdisk. Later we'll look at mounting a remote file-system over NFS to provide a diskless thin-client.