

Installing Debian/Ubuntu Over the Network

BV

[2015-03-28 Sat 11:32]

1 Overview

This summer we will have more students than ever before, maybe as many as 10. Last year we had five and managing the students workstations took substantial time and effort. I suspect this effort increases greater than linear with the number of students. The initial installation was just barely manageable in my "spare time". But, the tipping point is reached and I need something more efficient.

In the old Physics department co-op cluster, I used PXE to boot the worker nodes over the network. This entailed cooperation with a server listening for DHCP requests in order to dole out the usual initial IP configuration as well as to handle the extra PXE info like "next-server" and "filename" for the PXE bootloader that the machine should boot. The PXE bootloader then loaded Linux which mounted the root partition via NFS leaving the worker node effectively diskless.

The FAI project provides a system that works like this except instead of the boot ending with a ready-to-run OS it ends with the installation OS of your favorite Linux distribution (assuming your favorite is Debian or Ubuntu or some lesser Redhat based ones). The installation is preseeded with the answers to installation questions allowing for full installation to commence in a hands-off manner.

For the students workstations there is an extra challenge. With the co-op cluster the worker nodes and the DHCP/PXE server all lived on a non-routed private network. I could configure the server simply by logging in and tweaking its configuration file. In contrast, these workstations will be connected to the BNL internal network where the DHCP server is out of my hands and ITD reasonably does not want to maintain the very manual, detailed configuration that would be required to make it support PXE for my workstations.

Like most problems in this world, others have encountered them. Enter proxyDHCP where the PXE half of the communication is relegated to a second server independent from the "real" DHCP server. Both servers answer the broadcast sent by the PXE-booting workstation which aggregates the response and goes to town. At least that's the theory. We'll see how it goes.

2 The Plan

The installation plan is to test this on my home network and look for anything that might be a problem if repeated on the BNL network.

- Use 192.168.1.x local wired network using an Asus RT-AC56U router running the Tomato firmware (v1.28). It provides the DHCP server.
- Use system "haiku" (Ubuntu 14.04) as server
- Use two old crappy laptops (Sony Vaio "halfwit" and Dell Inspiron, "crappydell")
- Use dnsmasq for proxyDHCP and TFTP and nothing else
- Grab MAC addresses with `tcpdump`
- Install Ubuntu 14.10 or 15.04, configuration details d.b.d.

3 Preparation

3.1 Server install

```
$ sudo apt-get install dnsmasq tcpdump fai-quickstart
$ sudo apt-get remove tftpd-hpa
$ sudo apt-get remove atftpd
```

We only want dnsmasq as a TFTP server. The first remove is to get rid of the one that is brought in by FAI and the second is to get rid of the one that is brought in by removing the first. WTF?

3.2 MAC capture

FAI suggests:

```
$ sudo tcpdump -qtel broadcast and port bootpc >/tmp/mac.list
00:22:19:dc:2e:bb (oui Unknown) > Broadcast, IPv4, length 590: 0.0.0.0.bootpc > 255.25
bc:ee:7b:8e:62:f8 (oui Unknown) > Broadcast, IPv4, length 342: router.home.bootps > 25
```

This is capturing "crappydell" but fails to capture "halfwit". Apparently it's Ethernet NIC is crapped out. But, I can at least grab its wireless MAC the device list on the Tomato router's web page.

```
sony 00:13:CE:B3:39:18 (wireless)
dell 00:22:19:dc:2e:bb (wire)
```

For now, just focus on the "crappydell".

3.3 Configuring dnsmasq

There is not an overabundance of information on how to do this but it boils down to editing the configuration file `/etc/dnsmasq.d/proxydhcp.conf`.

Some links: [this](#) and [this](#) and [this](#) and [this](#)

Add to `/etc/dnsmasq.conf`

```
# Ubuntu sets up to use self for resolving and resolv.conf lists
# localhost so avoid self reference
no-resolv
```

```
# Router is real DNS server
server=192.168.1.1
```

```
# limit what interface to listen on
interface=eth0
# probably partly redundant
listen-address=127.0.0.1,192.168.1.123
```

```
# load aux config chunks
conf-dir=/etc/dnsmasq.d
```

```
    /etc/dnsmasq.d/proxydhcp.conf
```

```
# log traffic info
log-dhcp
# Turn on the included TFTP server
enable-tftp
# Root directory for TFTP files
```

```

tftp-root=/var/lib/tftpboot
# PXE boot loader
dhcp-boot=pxelinux.0
# Act as proxyDHCP on given network
dhcp-range=192.168.1.0,proxy
# Ignore anybody we don't know
dhcp-ignore=tag:!known
# last arg is pxe file bootloader sans ".0"
pxe-service=x86PC, "Boot PXELinux", pxelinux

    /etc/dnsmasq.d/known-hosts.conf

dhcp-host=00:22:19:dc:2e:bb

```

Put all known hosts in this file. Any not listed will simply be ignored by dnsmasq. Any changes to the config files requires a restart

```
# service dnsmasq restart
```

Monitor dnsmasq with

```
# tail -f /var/log/syslog
```

```
# cp /usr/lib/syslinux/pxelinux.0 /var/lib/tftpboot
```

This should be enough to get the one target host - and only that one host - booting the `pxelinux` bootloader. It will then fail as no PXE configuration is yet set up.

4 Apt proxy

```

# apt-get install approx
# mkdir /srv/approx # put it on a bigger disk.
# chown approx.approx /srv/approx

```

Configure proxy sources by first scoping out what a live system is using:

```
$ cat /etc/apt/sources.list /etc/apt/sources.list.d/* | grep -E "^[^#]" | cut -d " "
```

Edit `/etc/approx/approx.conf`

```
$cache          /srv/approx
```

```
ubuntu http://us.archive.ubuntu.com/ubuntu/  
ubuntu-security http://security.ubuntu.com/ubuntu  
ubuntu-extras http://extras.ubuntu.com/ubuntu  
ubuntu-mate http://ppa.launchpad.net/ubuntu-mate-dev/ppa/ubuntu  
ubuntu-mate-trusty http://ppa.launchpad.net/ubuntu-mate-dev/trusty-mate/ubuntu
```

Then, might as well change over to using it on the server.

5 FAI

5.1 Configuration space

Start with `/etc/fai/fai.conf`. The default "config space" is `/srv/fai/config` which will be NFS-exported to the subnet.

```
# echo '/srv/fai/config /nfs4/fai none bind,uid=1001 0 0' >> /etc/fstab  
# echo '/nfs4/fai 192.168.1.0/24(ro,sync,no_subtree_check)' >> /etc/exports  
# mkdir /nfs4/fai  
# mount /nfs4/fai  
# exportfs -a  
# showmount -e | grep fai  
/nfs4/fai      192.168.1.0/24
```

Test on some client:

```
$ sudo mount -t nfs haiku:/nfs4/fai /mnt/tmp  
$ ls /mnt/tmp  
class debconf disk_config files hooks package_config scripts tests  
$ sudo umount /mnt/tmp
```

Then set in `/etc/fai/fai.conf`

```
FAI_CONFIG_SRC=nfs://haiku/nfs4/fai
```

5.2 Config files

Edit:

- `/etc/fai/sources.list` to add the `approx apt-proxy/mirror` instead of hitting `ubuntu.com`.

- `/etc/fai/fai.conf` to set config space NFS URL
- `/etc/fai/make-fai-nfsroot.conf` for various things including which release to target and to use approx mirror (`FAI_DEBOOTSTRAP`)

5.3 Make NFS-root

```
# fai-setup -v
```

```
...
```

Shadow passwords are now on.

You can log into install clients without password using `/home/bv/.ssh/id_rsa.pub`

`cp: cannot stat '/home/sys/var/lib/fai/nfsroot/live/filesystem.dir/boot/vmlinu?-*': No`

`cp: cannot stat '/home/sys/var/lib/fai/nfsroot/live/filesystem.dir/boot/initrd.img-*':`

DHCP environment prepared. If you want to use it, you have to enable the `dhcpd` and the

`ERROR: No initrd installed.`