



# UbuntuOnCluster

---

This page documents the process of automatically installing ubuntu on a cluster of machines. This means one machine will be setup manually as install server and all others will be installed when booting. The version used here is hoary. As installer I used only the debian-installer (The default Ubuntu installer). So no tricks like FAI or kickstart are needed.

Prerequisites on the server:


- Ubuntu-base install
- An ubuntu  .iso
- nat active on the server  example or all machines must receive a public IP address by dhcp
- Several packages need to be installed

## Stage 1: Preparing DHCP & PXE booting

---



First you will have to know the mac addresses of all machines, so they will get unique and constant IP addresses and hostnames. For me this was easy since there was a list of them available.

Now install the packages `dhcp3-server` and `tftpd-hpa`

Once you have this list, you can edit your `/etc/dhcp3/dhcpd.conf`, an example can be found  here; it's self-explaining.

PXE booting requires that the .iso file is mounted locally, I mounted it under `/var/lib/tftpboot/ubuntu/`

```
mkdir /var/lib/tftpboot/ubuntu
echo '/data/ubuntu-5.04-install-i386.iso /var/lib/tftpboot/ubuntu/ auto
defaults,loop 0 0' >> /etc/fstab
mount -a
```

Next step is setting up the PXE config. I created two files: one for installing and one for booting from the local disk (ie: booting the installed system. Create `/var/lib/tftpboot/pxelinux.cfg` and put these files there:  default  localboot

As you can see, the default action is to run the installer. You can also set a few options in here (the kernel command line) To save some space on the kernel command line (space is

limited), create symlinks to relevant files:

```
cd /var/lib/tftpboot
ln -s ubuntu/install/netboot/ubuntu-installer/i386/initrd.gz
ubuntu/install/netboot/ubuntu-installer/i386/linux
ubuntu/install/netboot/pxelinux.0
ubuntu/install/netboot/ubuntu-installer/
```

You can see that in the example config files these symlinks are used.

## Stage 2: Setting up nis and nfs


---

For cluster machines, nis and nfs are usually used to share login information and parts of the filesystem. So you need to install both on the server. You will need the packages `nis` `nfs-kernel-server`.

Note: the `nis` package (not `nis` itself) is quite buggy, it will try to start `ypbind` even though you did not tell it to. It also completely ignores `presecd`, so in a following step we will create a new version of this package. For now it will do, you just have to wait a bit for `ypbind` to time out.

When the `nis` setup asks for a domain, pick one you like. As soon as `ypbind` times out, stop `nis` again with

```
invoke-rc.d nis stop
```

Now you need to edit `/etc/default/nis` and enable the `nis` server  (example). You also need to initialize the `nis` database with

```
/usr/lib/yp/ypinit -m
```

You can now start the `nis` services again

```
invoke-rc.d nis start
```

For NFS, you need to edit `/etc/exports` in order to export required parts of the filesystem. On my cluster I chose to export `/home` and `/data`, so the `exports` file looks like

```
# /etc/exports: the access control list for filesystems which may be
exported
#
to NFS clients. See exports(5).
/home 192.168.0.0/255.255.0.0(rw,async)
/data 192.168.0.0/255.255.0.0(rw,async)
```

Now restart the nfs server.

```
invoke-rc.d nfs restart
```

### Stage 3: Setting up local mirror and proxy

---

Installing from a local mirror and using an http proxy for the rest greatly improves the speed of subsequent installs. Since apt-proxy is quite broken, I chose to use squid as a generic http proxy. For the recovering the reboot part I also needed php, so I installed that too. (A CGI script would have worked just fine here, but i'm more familiar with php)

You will need the packages: `apache2 libapache2-mod-php4 squid`

My cluster has only one external IP address, so only the master server is connected to the internet. The other machines are connected only to the master (and via NAT they can reach the net). Because I don't want to be a public proxy, I told the squid installer to only listen on eth1 (the internal interface). You can tell apache to do so to by editing `/etc/apache2/ports.conf`.

```
invoke-rc.d apache2 stop
echo 'Listen 192.168.0.1:80' > /etc/apache2/ports.conf
echo 'Listen 127.0.0.1:80' >> /etc/apache2/ports.conf
invoke-rc.d apache start
```

You should also edit the squid config. An  example can be found on my homepage.


Now make the apache server an ubuntu archive by symlinking into the .iso

```
ln -s /var/lib/tftpboot/ubuntu/ubuntu /var/www/ubuntu
```

### Stage 4: preseed

---

Having everything in place on the server, we can now take care of the client configuration. The tftpboot will launch the ubuntu installer. This installer usually asks questions, but the answers can be preseeded in a so called preseed file.

A preseed file that answers all default questions and installs ubuntu-base and openssh-server can once again be found on my homepage  here.

Points for possible changes: language, package selection (which is an aptitude pattern), first-created user account and partition manager setup. Go over these settings and adapt them to your needs.

## Stage 5: Surviving the reboot

---

The installer reboots after the basic install, which means that the installer will be launched again. Of course you don't want this, which is why I created a so called registration system. As you can see in the preseed file, the preseed/late-command has been set to `wget http://192.168.0.1/register.php`. This does nothing on the client side, but the php script creates a PXE boot file for this machine which instructs it to boot from the local drive. If you want to reinstall a certain machine all you have to do is remove the associated PXE boot file and it will use the default again.

In order for this to work, the `www-data` user must have write access to `/var/lib/tftpboot/pxelinux.cfg`

```
chown :www-data /var/lib/tftpboot/pxelinux.cfg
chmod g+w $_
```

This is the `register.php` script:

```
<?php
function _dechex($x) { return sprintf("%02s",dechex($x)); }
$host =
strtoupper(implode('',array_map( dechex,explode('.', $ SERVER['REMOTE_ADDR'])))
    copy("/var/lib/tftpboot/pxelinux.cfg/localboot",
"/var/lib/tftpboot/pxelinux.cfg/$host");
?>
```

## Stage 6: postinstall

---

The postinstall script can be used to do anything you like. I used it to install a correct `nis` package on the clients, create a correct NIS config and more bootstrapping things.

Creating a correct NIS package is done as follows (for breezy this won't be necessary: the package has been fixed).

```
mkdir nispackage
cd nispackage
apt-get source nis
cd nis-3.12/debian
```

- Open the file `postinst`
- comment out line 61 (the one with `db_input`
- just before line 64 (the one with `if [ "$RET" add a line containing RET=domain`

where domain is the NIS domain you chose

- comment out line 106 (The one with db\_text)

Now enter the following sequence of commands:

```
cd ..
apt-get build-dep nis
dpkg-buildpackage
```

Copy the newly generated .deb (to be found in the nispackage folder) file to a location where the clients can retrieve them (either with wget or an nfs mount)

The postinstallscript itself should be placed in /var/www Mine looks like this:

```
#!/bin/bash

# Step 1: Initial mount
mkdir /data
mount -t nfs 192.168.0.1:/data /data -o
rw,soft,bg,rsize=32768,wsiz=32768,tcp,timeo=600,intr

# Step 2: Install and configure NIS
aptitude -y install portmap libslp1
dpkg -i /data/nis_3.12-3_i386.deb
cp /data/nsswitch.conf /etc/nsswitch.conf
echo '+:::::::' >> /etc/passwd
echo '+:::::::' >> /etc/shadow
echo '+:::' >> /etc/group

# Repair failing boot sequence (nis with nfs error)
DIR=`pwd`
cd /etc/rc2.d
ln -s ../init.d/mountnfs.sh S20mountnfs.sh
cd $DIR

# Step 2: Correct mounting
umount /data
echo 'enterprise:/home /home nfs
rw,soft,bg,rsize=32768,wsiz=32768,tcp,timeo=600,intr' >> /etc/fstab
echo 'enterprise:/data /data nfs
rw,soft,bg,rsize=32768,wsiz=32768,tcp,timeo=600,intr' >> /etc/fstab
mount -a
```

```
# Step 3: Correct sources.list and update
cp /data/sources.list /etc/apt/sources.list
aptitude update
aptitude -y upgrade

# Step 4: Installing other things and remove the installer log which
contains a cleartext password
aptitude -y install linux-686-smp manpages-dev
rm /var/log/installer/debconf-seed
rm /var/log/installer/cdebconf/questions.dat

# Step 5: Enable scheduled scripts
echo '0 * * * * root test -x /data/upgrade && /data/upgrade' >>
/etc/crontab

# Step 6: reboot, we installed a new kernel
reboot
```

Downloads:  [example nsswitch.conf](#)  [sources.list with local mirror](#)  [the post-install script](#)

CategoryDocumentation CategoryCleanup

l'ultima modifica è del 2006-12-11 21:47:40, fatta da Kso3