

Technical White Paper

Automated deployments of Ubuntu

By Nick Barcet – September 2008

© Copyright Canonical 2008



Overview

Whether you run a large data centre, have a few servers or manage desktop computers, it is often necessary for a system administrator to be able to automate the deployments of computers. Ubuntu offers multiple options to meet this need.

This white paper explains the general principles of automated deployment of Ubuntu and provides a few examples of common cases. It will also help answer one or more of the following questions:

- How can I re-install a server quickly in a disaster recovery scenario?
- Can I minimise the risk of human error when installing multiple computers?
- What can be done to automatically adapt the load of my servers in my virtual server farm?
- I am used to Red Hat's Kickstart, Novell's SuSE Autoyast, Sun's Jumpstart, Microsoft WDS, is there a similar tool for Ubuntu?

The goal of this white paper is to provide you with an overview of the tools available on Ubuntu and detail the process of putting some of the most common scenarios in place.

Revision History

- First release 2008-09-10
- Second release 2009-01-26
 - link updates
- Third release 2010-03-22
 - minor corrections, thanks to Yannick Cadin and Rainer Rohde

Table of Contents

Overview.....	2
Revision History.....	2
Introduction.....	5
Features and benefits of automated deployment in Ubuntu.....	5
Typical scenarios.....	7
Network deployments.....	7
Remote location deployments.....	7
Custom automated CD.....	7
Configuration methods.....	8
Debian Installer preseeds.....	8
Kickstart configuration.....	8
Recommended method.....	9
Setting up automated deployment	10
Planning your deployments.....	10
Preparing the configuration server.....	11
Configuring DHCP.....	11
Configuring DNS.....	12
Setting up TFTP.....	13
Preparing the Kickstart configuration.....	14
Initial setup using the Kickstart GUI.....	14
Adapting the configuration manually.....	15
Adding preseeds.....	15
Writing the installation scripts.....	15
Update boot defaults.....	16
Setting up a local mirror.....	17
Preparing for the installation of a new computer.....	19
Finding the computer MAC address.....	19
Updating DHCP information.....	19
Booting in PXE mode.....	19

Making an automated install CD.....	20
Appendix A – References and useful links.....	21
Appendix B – Ubuntu's Kickstart differences from RedHat implementation.....	22
Appendix C – Planning cue sheet.....	24

Introduction

When the number of computers in an organisation increases, the need to use a tool to easily deploy additional servers rises in order to reduce the overall cost of maintaining these systems. Many use tools such as ghosting solutions to provide this functionality, but they are quite limited by the fact that they will only work on nearly identical hardware and therefore need constant attention in order to be kept up-to-date with new hardware.

Ubuntu, whether it is used for the desktop or server, provides the tools needed to pre-configure the official installer and therefore allows for reproducible, hardware-independent deployment scenarios that can be easily adapted to match virtually every need. Because of the nature of Linux, this functionality is not limited to the operating system itself. It also naturally encompasses the automated deployment and configuration of any application for which a package exists, and if an application is not yet packaged, it is equally possible to script its deployment manually.

Features and benefits of automated deployment in Ubuntu

Feature	Benefit
Network deployment	Allows control of the automated deployment on multiple machines over the network remotely, defining their roles centrally and ultimately reassigning them. This reduces the physical intervention of plugging new machines into the network.
CD deployment	Facilitates the automated deployment of machines that are not within the normal network reach, with the only physical intervention being the need to plug in a machine and insert a CD.
Unattended installation	Once a new computer is switched on installation can be done in an unattended manner without any user intervention.
Controlled installation	Automated installations can include the request for a user to enter some information, allowing, for example, easy site configuration.
Pre and post flight scripting	Allows for specific actions to take place before any installation has started and after it has completed, therefore allowing full control of the complete process.
Easy configuration GUI	An easy graphical tool is available in Ubuntu that allows definition of the most common installation parameters and the automated generation of the corresponding control file.
Automated updates	Ubuntu provides all the means to define automated installation of updates on a regular basis.

Landscape integration	Computers supported through a contract with Canonical can be automatically integrated to the Landscape systems management framework. Therefore confirming the proper installation, update and full life cycle of every machine on a network
------------------------------	---

Typical scenarios

Network deployments

The most common use of an automated deployment is in a data centre. It would be used to quickly add new servers, or reconfigure an existing one while minimising the risk for errors and providing all the base elements that are required. This is the usual method to deploy racks of machines or blade systems over-the-wire. It can also be used to deploy desktops.



Figure 1 - Automated deployment

To do this, it is customary to setup one machine as a DHCP/TFTP/HTTP server that provides the configuration information for installation over the network. To be installed the server simply has to boot in PXE mode and the installation process starts and completes in the background, retrieving all the the requested packages from a repository server. The repository server can be a local mirror or one of the official Ubuntu mirrors available throughout the the world.

Remote location deployments

Similar in principle to the data centre deployments, deploying from a remote location allows the same flexibility but requires the network to be configured to provide DHCP relays on the remote locations that redirect the first DHCP requests to a central location.

Custom automated CD

A less common but quite useful use of automated deployment is making a custom installation CD sufficient to perform the full installation. This technique is quite handy in situations where multiple servers have to be deployed in remote geographical locations with no or limited connectivity.

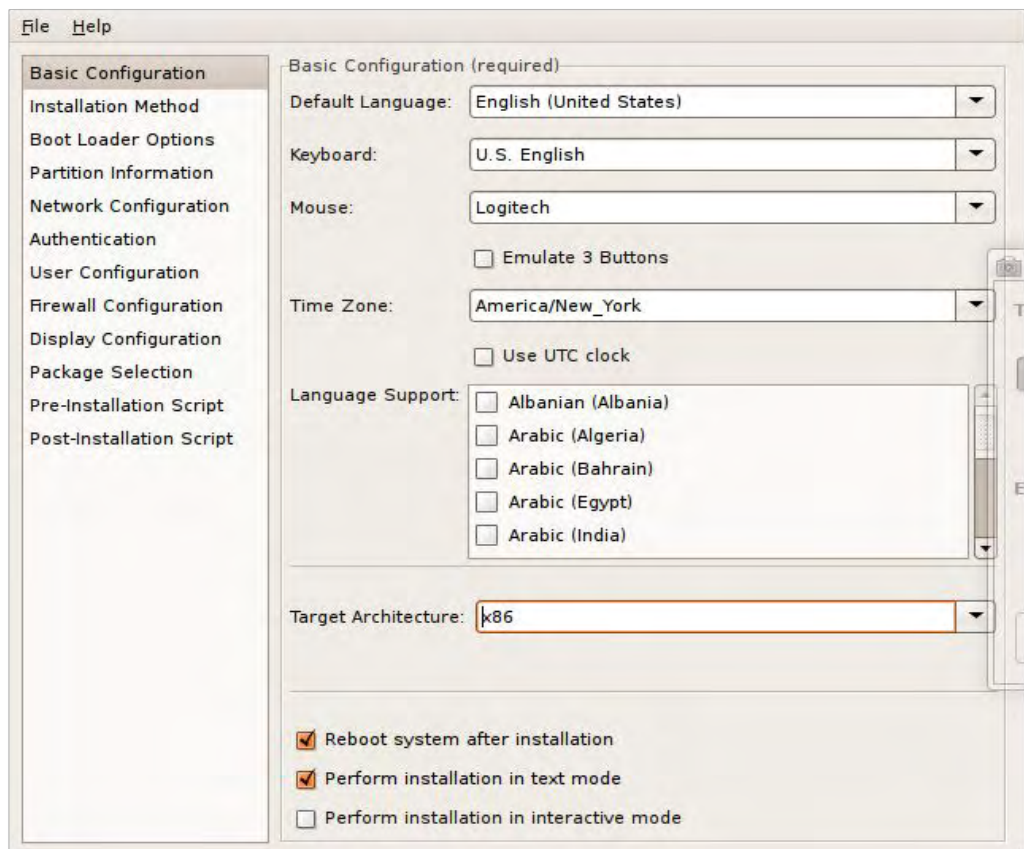
Configuration methods

Debian Installer preseeds

As a Debian derivative, Ubuntu Linux uses the Debian Installer during the install process of our operating system. Experienced Debian users will instinctively point you to the preseeds configuration that the Debian Installer can read to answer the questions it would normally ask of the system administrator. While it is certainly the most comprehensive method to automate an installation process, it is also the most complex method to use if you have to define your entire configuration.

Kickstart configuration

As an alternative, Ubuntu implements its own version of Kickstart, which is not only less complex, but also offers a graphical interface to create your initial configuration file. Kickstart was originally developed for Red Hat and while great care was taken to port it to Ubuntu, a few functionalities could not be implemented, so it is not as complete as preseeds is (please see appendix A for details of the functionalities).



Recommended method

The good news is that both preseeds and Kickstart can be combined in one single installation, doing the base work using Kickstart and completing it using the preseeds command conveniently added to the Ubuntu's version of Kickstart. You could even specify a complete preseeds configuration file separately to the Kickstart file. While Kickstart will satisfy 90% of the requirements, the preseeds will allow the fine tuning that is sometimes necessary and not available through Kickstart. The remainder of this paper describes how to set up your infrastructure to make sure you cover 100% of your requirements.

Setting up automated deployment

Planning your deployments

The very first step in order to set up an automated deployment scenario is to write down your requirements so that you ensure you will not forget anything later. The following cue sheet can help you plan it better. We will fill in the cue sheet to cover our specific scenario in this white paper as an example. You can find a blank cue sheet in Appendix C.

Type of deployment	Web Server															
Purpose	Automated network based LAMP server deployment in farm.															
Requirements	<ul style="list-style-type: none"> • Adding a new server, should be limited to updating DHCP and DNS configuration and starting the server • Apache should be configured with PHP and Python • MySQL should be configured with standard password • Landscape client should be installed and configured by default 															
Disk	Main disk should be partitioned as: <table border="1" data-bbox="687 1106 1380 1440"> <thead> <tr> <th>Mount point</th> <th>Type</th> <th>Size (M)</th> </tr> </thead> <tbody> <tr> <td>/</td> <td>Ext3</td> <td>2048</td> </tr> <tr> <td>/boot</td> <td>Ext3</td> <td>512</td> </tr> <tr> <td>/var</td> <td>Ext3</td> <td>Remaining space</td> </tr> <tr> <td>Swap</td> <td>Swap</td> <td>1024</td> </tr> </tbody> </table>	Mount point	Type	Size (M)	/	Ext3	2048	/boot	Ext3	512	/var	Ext3	Remaining space	Swap	Swap	1024
Mount point	Type	Size (M)														
/	Ext3	2048														
/boot	Ext3	512														
/var	Ext3	Remaining space														
Swap	Swap	1024														
Network	Device: eth0 Address: DHCP fixed by MAC address															
X Window System	-															
Packages	ubuntu-standard lamp-server openssh-server unattended-upgrades landscape-client															
Pre install. script	-															
Post install. script	landscape configuration															

Preparing the configuration server

For this step we will install a new Ubuntu server that will act as a DNS, DHCP, TFTP and HTTP server.

Note: Before installing a DHCP server, you should always make sure that no other DHCP servers already exist on the network you are doing your installation on, or a conflict will occur. If unsure, please consult the network administrator.

- Make a manual installation of Ubuntu Server Edition, making sure that you assign a fixed address to your machine, because DNS and DHCP server requires to have a fixed address. If there is no DHCP server on the network, the installer should automatically prompt you to manually specify an address, otherwise using the “Go Back” function in most dialogs will bring you back to the menu so you can specify an address manually
- The last step of the installer should prompt you to preconfigure tasks: simply select DNS Server (and openssh-server if you want to access this server remotely).
- Once base installation is complete, the following step installs the other packages that we will need. Use your favourite install command to add the following list of packages:

- openbsd-inetd
- tftpd-hpa
- apache2
- dhcp3-server
- unattended-upgrades

Configuring DHCP

Edit the base configuration of your DHCP server as you wish so that it is ready to answer requests from the network it is connected to. In this sample environment, the resulting `/etc/dhcp3/dhcpd.conf` file ends up as follows, but it should be adapted to your specific environment:

```
# configure dynamic dns update
ddns-update-style interim;
ddns-domainname "canonical.lan";
ddns-rev-domainname "in-addr.arpa";
update-static-leases on;
zone canonical.lan. {
    primary 192.168.1.6; #update the local server
}
zone 0.168.192.in-addr.arpa. {
    primary 192.168.1.6; #same for reverse addresses
}

# global options
option domain-name "canonical.lan";
option domain-name-servers 192.168.1.6;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;
```

```
#specify the TFTP server address
next-server 192.168.1.6;

# Subnet for workstations
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.150 192.168.1.200;
    option broadcast-address 192.168.1.255;
    option routers 192.168.1.2;
}
```

In the above, you will need to change:

- “canonical.lan” for the DNS domain name used on your network
- 192.168.1.6 for the IP address of your DNS server
- 192.168.1.2 for the IP address of the default gateway on your network

Configuring DNS

Edit the local configuration of your DNS server as you wish so that updates are allowed to be provided by the DHCP server. In this sample environment, here are the resulting files after editing:

/etc/bind/named.conf.local

```
zone "canonical.lan" {
    type master;
    file "/etc/bind/db.canonical.lan";
    allow-update {
        127.0.0.1;
        192.168.1.6;
    };
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192.168.1";
    allow-update {
        127.0.0.1;
        192.168.1.6;
    };
};
```

/etc/bind/db.canonical.lan

```
$TTL 1d
@      IN      SOA      master.canonical.lan. root.canonical.lan. (
    2007122001      ; Serial
    24H             ; Delay
    2H              ; Sup Delay
    1000H           ; Obsolete
    2D)             ; Cache

canonical.lan.    IN      NS       master.canonical.lan.
master            IN      A        192.168.1.6
```

/etc/bind/db.192.168.1

```
$TTL 1D
```

```
@      IN      SOA      master.canonical.lan.  root.canonical.lan.  (  
      2007122001      ; Serial  
      24H      ; Delay  
      2H      ; Sup Delay  
      1000H     ; Obsolete  
      2D)     ; Cache  
  
@      IN      NS      master.canonical.lan.  
6      IN      PTR     master.canonical.lan.
```

Then you need to modify the permissions on the two db files so that the DNS process, running as user “bind” can update them:

```
sudo chown bind /etc/bind/db.192.168.1 /etc/bind/db.canonical.lan  
sudo chmod 664 /etc/bind/db.192.168.1 /etc/bind/db.canonical.lan
```

Then we just have to restart our DNS server using: `sudo /etc/init.d/bind9 restart`.

Setting up TFTP

To get the TFTP server ready to go, you should first make sure that `tftpd` is enabled. It is enabled by the following line in `/etc/inetd.conf`:

```
tftp dgram  udp wait      root  /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s  
/var/lib/tftpboot
```

This should have been set up automatically if you have installed `openbsd-inetd` before `tftphpa`.

Read that line and remember the directory which is used as the argument of `in.tftpd`; you'll need that below (here `/var/lib/tftpboot`). If you've had to change `/etc/inetd.conf`, you'll have to notify the running `inetd` process that the file has changed, so run:

```
sudo /etc/init.d/inetd reload.
```

You then need to put the TFTP boot image in place. These images are found in the official Ubuntu mirror within the `/main/installer-{arch}/current/images/` of any given version. You need the `netboot.tar.gz` file in this directory. You will download and extract it within the subdirectory of `tftpboot` directory that we identified earlier:

```
cd /var/lib/tftpboot/  
sudo mkdir webserver  
cd webserver  
sudo wget http://archive.ubuntu.com/ubuntu/dists/hardy/main/installer-i386/current/images/netboot/netboot.tar.gz  
sudo tar -xzvf netboot.tar.gz
```

Notice the file `pxelinux.0` that has been extracted, it is what you will point our client to when you will configure dhcp for tftp booting. Notice the file `pxelinux.cfg/default` as you will modify it later to specify the configuration files.

Preparing the Kickstart configuration

Initial setup using the Kickstart GUI

On your Ubuntu desktop machine, install the package `system-config-kickstart` using your favourite tool. You will then find the tool in Applications->System Tools->Kickstart. Use this application to create your initial `ks.cfg` file according to your cue sheet.

Note: do not use `admin` or any system user name as your default account or installation will be interrupted.

In this case, you end up with the following file:

```
#Generated by Kickstart Configurator
#platform=x86

#System language
lang en_US
#Language modules to install
langsupport fr_FR --default=en_US
#System keyboard
keyboard us
#System mouse
mouse
#System timezone
timezone Europe/Paris
#Root password
rootpw --disabled
#Initial user
user first --fullname "myuser" --iscrypted --password
$1$N/OJDloB$/kIgbwxuSICTaspBGAdLH.
#Reboot after installation
reboot
#Use text mode install
text
#Install OS instead of upgrade
install
#Use Web installation
url --url http://archive.ubuntu.com/ubuntu
#System bootloader configuration
bootloader --location=mbr
#Clear the Master Boot Record
zerombr yes
#Partition clearing information
clearpart --all --initlabel
#Disk partitioning information
part / --fstype ext3 --size 2048
part /boot --fstype ext3 --size 512
part /var --fstype ext3 --size 1 --grow
part swap --size 1024
#System authorization information
auth --useshadow --enablemd5
#Network information
network --bootproto=dhcp --device=eth0
#Firewall configuration
firewall --disabled
#Do not configure the X Window System
skipx
```

```
#Package install information
%packages
@ ubuntu-desktop
%pre
#this is my pre installation script
%post
#this is my post installation script
```

Adapting the configuration manually

The configuration still needs to be adapted slightly so that it will match our requirements. The first thing to do is adapt the package list so the package section now looks like:

```
#Package install information
%packages
@ lamp-server
@ openssh-server
unattended-upgrades
```

Notice that tasks are specified with a '@'.

Adding preseeds

Next you need to preseed the question for the mysql password. You first need to know what the question "keyword" is. To do this, you will need a machine with mysql is installed. Then look in the `/var/lib/dpkg/info/` directory for a mysql-server file suffixed by `.templates`. There is only one such file in there and opening it will quickly identify the question you are looking for by searching for the occurrence of the word "password":

```
Template: mysql-server/root_password
Type: password
Description: New password for the MySQL "root" user:
```

You can now add the following command to our Kickstart file just before the packages section:

```
preseed mysql-server/root_password password mydefaultpassword
```

Writing the installation scripts

Installing the landscape client requires you to add a new source for apt, and since the package is signed, you also need to add the key to the apt key chain. This is something that you'll need to do in the post-installation phase before you can install it.

Configuring landscape-client is a bit more complicated as it will require you to have a fully configured system and this will not be the case until after your server has rebooted once. In order to solve this you will need to modify the `/etc/rc.local` script in your post install script so that it calls your landscape initialization script once after the first boot.

Your post installation script will look as follows:

```
# modify rc.local so that it call our script once
```

```
mv /etc/rc.local /etc/rc.local.orig
cat > /etc/rc.local <<EOT
#!/bin/sh -e
#execute initial-config only once
if [ ! -e /root/initial_config_done ]; then
    /root/initial_config && touch /root/initial_config_done
fi
exit 0
EOT
chmod a+x /etc/rc.local

# create the script that will be executed at first boot
cat > /root/initial_config <<EOT
exit
gpg --keyserver-options http-proxy --keyserver keyserver.ubuntu.com \
    --recv-key C605E80D
gpg --armor --export C605E80D | apt-key add -
apt-get update
apt-get -y install landscape-client
configure-landscape --computer-title "$(cat /etc/hostname)" \
    --account-name user \
    --registration-password password
mv /etc/rc.local.orig /etc/rc.local
EOT
chmod a+x /root/initial_config
```

Note: the username and password will have to change to the value you want in the above example.

Note: you can, instead using the command `apt-get -y landscape-client` in the post install script, use the following preseed in the kickstart script: `preseed pkgsel/include string landscape-client`

Update boot defaults

Now that your configuration file is ready, you should now save it on your web server. Any location and name would be fine, but in this case you should place it as `/var/www/ks.cfg`, so that it will be accessible as <http://192.168.1.6/ks.cfg>.

Because you are installing a server, you need to retrieve the seed that is used on the server CD Rom under `/preseed/ubuntu-server.seed` and place it on the server as well.

Finally you need to update the boot defaults in your tftpboot directory so that the installer is instructed to use this file. Modify the file

`/var/lib/tftpboot/webserver/pxelinux.cfg/default` to add a new LABEL as follows :

```
LABEL auto
    kernel ubuntu-installer/i386/linux
    append vga=normal preseed/url=http://192.168.1.6/ubuntu-server.seed
ks=http://192.168.1.6/ks.cfg initrd=ubuntu-installer/i386/initrd.gz --
```

Note: the append arguments end with a double dash (--). Any arguments before this double dash will only be used to boot the system during installation, while arguments that would be passed after it would also be used to boot the installed system.

Next, modify the `DEFAULT` parameter to point to `auto`.

Finally change the `TIMEOUT` to 1, so that installation will start after a 1 second wait

Note: To add more preseed directives to the `ubuntu-server.seed` file, see Appendix A for a link to the preseed configuration manual.

Security Note: As you certainly noticed, these files contain password written in clear text, so they become available to anyone with read access to them. It is therefore imperative to make sure that access to them is limited as much as possible and that your network is secure. It would also be a good practice to change these default passwords as soon as possible.

Setting up a local mirror

If you plan to set up multiple machines, or if you want your installations to be a bit faster, it might be a good idea to set up a local mirror of the Ubuntu repositories for the targeted architectures that you plan to support. The package `apt-mirror` provides you with a script that will handle the mirroring for you. You should plan on having about 20 gigabytes of free space per supported release and architecture. There are quite a few other tools that can provide similar services, each with its own sets of features, `apt-mirror` is just one of these tools, not the one I would absolutely recommend.

By default, `apt-mirror` uses the configuration file in `/etc/apt/mirror.list`. As it is set up, it will replicate only the architecture of the local machine. If you would like to support other architectures on your mirror, simply duplicate the lines starting with “`deb`”, replacing the `deb` keyword by `/deb-{arch}` where `arch` can be `i386`, `amd64`, etc... For example, on an `amd64` machine, to have the `i386` archives as well, you will have:

```
deb http://archive.ubuntu.com/ubuntu hardy main restricted universe multiverse
/deb-i386 http://archive.ubuntu.com/ubuntu hardy main restricted universe
multiverse
```

```
deb http://archive.ubuntu.com/ubuntu hardy-updates main restricted universe
multiverse
/deb-i386 http://archive.ubuntu.com/ubuntu hardy-updates main restricted
universe multiverse
```

```
deb http://archive.ubuntu.com/ubuntu/ hardy-backports main restricted universe
multiverse
/deb-i386 http://archive.ubuntu.com/ubuntu hardy-backports main restricted
universe multiverse
```

```
deb http://security.ubuntu.com/ubuntu hardy-security main restricted universe
multiverse
/deb-i386 http://security.ubuntu.com/ubuntu hardy-security main restricted
universe multiverse
```

```
deb http://archive.ubuntu.com/ubuntu hardy main/debian-installer
restricted/debian-installer universe/debian-installer multiverse/debian-
installer
/deb-i386 http://archive.ubuntu.com/ubuntu hardy main/debian-installer
restricted/debian-installer universe/debian-installer multiverse/debian-
installer
```

Notice that the source packages are not mirrored as they are seldom used compared to the binaries and they take up a lot more space. They can be easily added to the list.

Once the mirror has finished replicating (and this can be quite long), you need to configure Apache so that your mirror files (in `/var/spool/apt-mirror` if you did not change the default), are published by your Apache server. I am not going to detail this process here, as Apache configuration is not our goal and many tutorials already exist on the subject.

Then, go back to your `ks.cfg` file and modify the `url` parameter to point to your mirror. Remember that you only need to specify the Ubuntu root path. To have the security updates point to your local mirror, you will also need to add the following preseed command right below the url command, replacing `security.ubuntu.com` with the name of your host:

```
preseed apt-setup/security_host string security.ubuntu.com
```

Preparing for the installation of a new computer

Finding the computer MAC address

If you try booting a computer using PXE, the first thing that it normally displays is its Ethernet MAC address. Write it down as we are going to need this address for each computer that we will automatically deploy. The address has the form XX:XX:XX:XX:XX:XX.

Updating DHCP information

We now need to add to the `/etc/dhcp3/dhcpd.conf` file the information about our new computer. In our case the following:

```
host web1 {
    hardware ethernet 00:0C:29:EE:9D:E0;
    option host-name "web1";
    fixed-address 192.168.1.10;
    filename "/webserver/pxelinux.0";
}
```

Once you have finished your modifications, test them using the command `sudo dhcpd3 -t` and restart your DHCP server with `sudo /etc/init.d/dhcp3-server restart`.

Booting in PXE mode

All BIOS released since 1999 should now include the possibility to boot in PXE mode. It is generally placed in a boot menu. On most computers, when no operating system is installed, PXE boot will be tried by default after testing CDROM and hard disk and not finding anything to boot from. This means that in theory, all you have to do at this point is to plug in your computer to the network and power it on to start the installation process.

It is now time to test your automated installation. If something goes wrong, you should be able to determine what is the problem by looking at the log files you will find on the configuration server under `/var/log/`:

- `syslog`: will show information about tftp and dhcp requests
- `apache2/*`: will show logs of the requests made to the http server

Making an automated install CD

To make a CDROM that will automatically install Ubuntu, you'll first need to mount the official CD you want to automate locally. For example, if you have downloaded the ISO image on your disk, you can mount this image using the command `mount` with the `loop` option:

```
sudo mount -o loop /home/nicolas/ubuntu-7.10-server-i386.iso /mnt
```

We then can copy the contents of the CD to a local directory of our choice (including dot files):

```
rm -rf dest #remove existing
cp -a /mnt dest #will create dest because of -a
```

Modify it so that we can add a Kickstart parameter to the `isolinux/isolinux.cfg` file:

```
LABEL install
  menu label ^Install to the hard disk automatically
    kernel /install/vmlinuz
    append file=/cdrom/preseed/ubuntu-server.seed initrd=/install/initrd.gz
vga=normal ks=cdrom:/isolinux/ks.cfg --
```

Note: In this case it is a bad idea to modify the `timeout` parameter or you risk wiping out working systems if the machine was involuntarily booted from this CDROM.

Note: the `append` arguments end with a double dash (`--`). Any arguments before this double dash will only be used to boot the system during installation, while arguments that would be passed after it would also be used to boot the installed system.

Now we can create our Kickstart file similarly to the case of a network install, except that:

- we will have a `cdrom` command instead of the `url` one
- to avoid the grub confirmation dialog at the end of the installation we'll preseed the question with:

```
preseed grub-installer/only_debian boolean true
```
- because `unattended-upgrades` is not available on the CD, we'll have to remove it from the list of packages (we cannot mix `cdrom` and network sources during an install), and alternatively either:
 - to have `unattended-upgrades` installed after the first reboot, we'll modify the `apt-get install` line of our script to: `apt-get -y install landscape-client unattended-upgrades`. We'll place it in the same `isolinux` directory.
 - Or add the following preseed to our kickstart script: `preseed pkgset/include string landscape-client unattended-upgrades`

Once our Kickstart file is ready, we can create a new ISO:

```
mkisofs -J -l -b isolinux/isolinux.bin -no-emul-boot -boot-load-size 4 -boot-info-table -z -iso-level 4 -c isolinux/isolinux.cat -o ./autoinstall.iso
newiso/
```

We're ready to burn and test the new autoinstall CD.

Appendix A – References and useful links

- Ubuntu preseed manual:
<http://help.ubuntu.com/8.04/installation-guide/i386/appendix-preseed.html>
- Ubuntu Automatic Installation manual:
<http://help.ubuntu.com/8.04/installation-guide/i386/automatic-install.html>
- Red Hat's Kickstart documentation (for RHEL 4, while Ubuntu's Kickstart implementation is based of RHL 9):
<http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/sysadmin-guide/pt-install-info.html>

Appendix B – Ubuntu's Kickstart differences from RedHat implementation

The following table details the extensions that were added for Ubuntu:

rootpw	<p>Now takes the <code>--disabled</code> option to disable the root password. If this is used, the initial user will be given root privileges via sudo.</p>
user	<p>A new <code>user</code> command has been added to control the creation of the initial user:</p> <pre>user joe [--fullname "Joe User" --password iamjoe</pre> <ul style="list-style-type: none"> • <code>--disabled</code> option prevents any non-root users from being created. • <code>--fullname</code> option specifies the user's full name, as opposed to the Unix username. • <code>--password</code> option supplies the user's password, by default in the clear (in which case make sure your Kickstart file is kept confidential!); • <code>--iscrypted</code> option may be used to state that the password is already MD5-hashed.
preseed	<p>Has been added to provide a convenient way to preseed additional items in the debconf database that are not directly accessible using the ordinary Kickstart syntax:</p> <pre>preseed [--owner login] package/question type value</pre> <ul style="list-style-type: none"> • <code>package/question</code> is the name of the question as defined in <code>/var/lib/dpkg/info/*.template</code> • <code>type</code> is the question type as defined in the same template file • <code>value</code> is the value that should be user • <code>--owner</code> option sets the name of the package that owns the question; if omitted, it defaults to <code>d-i</code>, which is generally appropriate for items affecting the first stage of the installer.
keyboard	<p>now takes <code>layout_variant</code> as X layout names. For example, <code>in_guj</code> selects the Gujarati variant of the Indian layout.</p>

The following table details the Kickstart configuration directives not implemented in Ubuntu:

Authentication	<ul style="list-style-type: none"> • LDAP, Kerberos 5, Hesiod, and Samba authentication. • (<code>--auth --enablecache</code>) command to enable nscd
LILO	<ul style="list-style-type: none"> • <code>bootloader --linear, --nolinear, and --lba32</code> options for detailed LILO configuration • <code>lilocheck</code> command to verify existing LILO configuration
Upgrade	<ul style="list-style-type: none"> • Ubuntu is designed to handle in-place upgrades using <code>apt</code> and its various front-ends (<code>aptitude, synaptic, etc...</code>) to provide this functionality without requiring the use of an installer
Partitioning	<ul style="list-style-type: none"> • Partitioning is limited to the main disk • Logical Volume Management (LVM) configuration • Checking a disk for bad sectors • RAID partitioning • Restrictions of a partition to a particular disk or device, and specifications of the starting or ending cylinder for a partition.
Drivers	<ul style="list-style-type: none"> • It is not possible to provide a driver disk during automated install • The <code>device</code> command is not available to define additional kernel module
Firewall	<ul style="list-style-type: none"> • None of the firewall configuration commands are supported
Installation source	<ul style="list-style-type: none"> • The method supported by Anaconda of adding a plain "ks" boot parameter to work out the location of the Kickstart file from a DHCP response is not yet supported by the Ubuntu installer. • NFS • local hard disk
X configuration	<ul style="list-style-type: none"> • <code>xconfig --monitor</code> option to use a specified monitor name.
Packages	<ul style="list-style-type: none"> • Package groups do not match the Red-Hat definition. They are translated to tasks in Ubuntu (use <code>taskse1 --list-tasks</code> to list available tasks in the current release) • Exclusions in <code>%packages</code> sections are not supported. You can use a <code>%post</code> script instead to remove unnecessary packages.
Scripts	<ul style="list-style-type: none"> • Scripts may only be shell scripts during pre and post installation phase.

Appendix C – Planning cue sheet

Type of deployment	Name here the type of install you will be doing for later reference																					
Purpose	Explain here the purpose of the installation																					
Requirements	<p>Explain here in a few bullet points which are the particularities you would like to achieve</p> <ul style="list-style-type: none"> • • • • 																					
Disk	<p>Main disk should be partitioned as:</p> <table border="1"> <thead> <tr> <th>mount point</th> <th>type</th> <th>size (M)</th> </tr> </thead> <tbody> <tr> <td>/</td> <td></td> <td></td> </tr> <tr> <td>/boot</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table>	mount point	type	size (M)	/			/boot														
mount point	type	size (M)																				
/																						
/boot																						
Network	specify here how should the network be configured																					
X Window System	specify here if and how should X be configured																					
Packages	list here all packages that should be installed																					
Pre install script	what should the pre install script do?																					
Post install script	what should the post install script do?																					

Every effort has been made by Canonical to ensure the accuracy of this document but Canonical disclaims, to the extent possible at law, any liability for any error or omission.

© Canonical Limited 2008. Ubuntu and associated logos are registered trademarks of Canonical Ltd., all rights reserved. All other trademarks are the properties of their respective owners. Any information contained in this document may change without notice and Canonical is not held responsible for any such changes.