

STEP BY STEP SERVER LINUX UBUNTU

By Wahyu Subagiyo, Skom

After the reboot you can login with your previously created username (e.g. administrator). Because we must run all the steps from this tutorial as root user, we must enable the root account now.

Run

```
-----  
sudo passwd root  
-----
```

and give root a password. Afterwards we become root by running

```
-----  
su  
-----
```

5 Install The SSH Server (Optional)

If you did not install the OpenSSH server during the system installation, you can do it now:

```
-----  
apt-get install ssh openssh-server  
-----
```

From now on you can use an SSH client such as [PuTTY](#) and connect from your workstation to your Ubuntu 8.04 LTS server and follow the remaining steps from this tutorial.

6 Install vim-full (Optional)

I'll use vi as my text editor in this tutorial. The default vi program has some strange behaviour on Ubuntu and Debian; to fix this, we install vim-full:

```
-----  
apt-get install vim-full  
-----
```

(You don't have to do this if you use a different text editor such as joe or nano.)

7 Configure The Network

Because the Ubuntu installer has configured our system to get its network settings via DHCP, we have to change that now because a server should have a static IP address. Edit `/etc/network/interfaces` and adjust it to your needs (in this example setup I will use the IP address `192.168.0.100`):

vi /etc/network/interfaces

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.0.100
    netmask 255.255.255.0
    network 192.168.0.0
    broadcast 192.168.0.255
    gateway 192.168.0.1
```

Then restart your network:

/etc/init.d/networking restart

Then edit /etc/hosts. Make it look like this:

vi /etc/hosts

```
127.0.0.1    localhost.localdomain localhost
192.168.0.100 server1.example.com  server1
# The following lines are desirable for IPv6 capable hosts
::1    ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Now run

```
echo server1.example.com > /etc/hostname
/etc/init.d/hostname.sh start
```

Afterwards, run

```
hostname
hostname -f
```

Both should show server1.example.com now.

8 Edit /etc/apt/sources.list And Update Your Linux Installation

Edit /etc/apt/sources.list. Comment out or remove the installation CD from the file and make sure that the universe and multiverse repositories are enabled. It should look like this:

vi /etc/apt/sources.list

```
#
# deb cdrom:[Ubuntu-Server 8.04 _Hardy Heron_ - Release i386 (20080423.2)]/ hardy
main restricted
#deb cdrom:[Ubuntu-Server 8.04 _Hardy Heron_ - Release i386 (20080423.2)]/ hardy
main restricted
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://de.archive.ubuntu.com/ubuntu/ hardy main restricted
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy main restricted
## Major bug fix updates produced after the final release of the
## distribution.
deb http://de.archive.ubuntu.com/ubuntu/ hardy-updates main restricted
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy-updates main restricted
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## universe WILL NOT receive any review or updates from the Ubuntu security
## team.
deb http://de.archive.ubuntu.com/ubuntu/ hardy universe
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy universe
deb http://de.archive.ubuntu.com/ubuntu/ hardy-updates universe
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy-updates universe
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://de.archive.ubuntu.com/ubuntu/ hardy multiverse
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy multiverse
deb http://de.archive.ubuntu.com/ubuntu/ hardy-updates multiverse
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy-updates multiverse
## Uncomment the following two lines to add software from the 'backports'
## repository.
## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes
## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.
# deb http://de.archive.ubuntu.com/ubuntu/ hardy-backports main restricted universe
multiverse
# deb-src http://de.archive.ubuntu.com/ubuntu/ hardy-backports main restricted
universe multiverse
## Uncomment the following two lines to add software from Canonical's
## 'partner' repository. This software is not part of Ubuntu, but is
## offered by Canonical and the respective vendors as a service to Ubuntu
## users.
```

```
# deb http://archive.canonical.com/ubuntu hardy partner
# deb-src http://archive.canonical.com/ubuntu hardy partner
deb http://security.ubuntu.com/ubuntu hardy-security main restricted
deb-src http://security.ubuntu.com/ubuntu hardy-security main restricted
deb http://security.ubuntu.com/ubuntu hardy-security universe
deb-src http://security.ubuntu.com/ubuntu hardy-security universe
deb http://security.ubuntu.com/ubuntu hardy-security multiverse
deb-src http://security.ubuntu.com/ubuntu hardy-security multiverse
```

Then run

apt-get update

to update the apt package database and

apt-get upgrade

to install the latest updates (if there are any).

9 Change The Default Shell

/bin/sh is a symlink to /bin/dash, however we need /bin/bash, not /bin/dash. Therefore we do this:

```
ln -sf /bin/bash /bin/sh
```

If you don't do this, the ISPCConfig installation will fail.

10 Disable AppArmor

AppArmor is a security extension (similar to SELinux) that should provide extended security. In my opinion you don't need it to configure a secure system, and it usually causes more problems than advantages (think of it after you have done a week of trouble-shooting because some service wasn't working as expected, and then you find out that everything was ok, only AppArmor was causing the problem). Therefore I disable it (this is a must if you want to install ISPCConfig later on).

We can disable it like this:

```
-----
/etc/init.d/apparmor stop
update-rc.d -f apparmor remove
-----
```

Till told me that he also had to do this step (which was not necessary on my installation), so if you want to go sure, do this on your system as well:

```
apt-get remove apparmor apparmor-utils
-----
```

11. Install Some Software

Now we install a few packages that are needed later on. Run

apt-get install binutils cpp fetchmail flex gcc libarchive-zip-perl libc6-dev libcompress-zlib-perl libdb4.3-dev libpcre3 libpopt-dev lynx m4 make ncftp nmap openssl perl perl-modules unzip zip zlib1g-dev autoconf automake1.9 libtool bison autotools-dev g++ build-essential

(This command must go into **one line!**)

12 Quota

(If you have chosen a different partitioning scheme than I did, you must adjust this chapter so that quota applies to the partitions where you need it.)

To install quota, run

```
apt-get install quota
```

Edit /etc/fstab. Mine looks like this (I added ,usrquota,grpquota to the partition with the mount point /):

```
vi /etc/fstab
```

```
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
# /dev/sda1
UUID=6af53069-0d51-49be-b275-aeaea8d780c5 / ext3
relatime,errors=remount-ro,usrquota,grpquota 0 1
# /dev/sda5
UUID=d8e1f66c-1442-423e-b442-8ae66eded9d7 none swap sw 0
0
/dev/scd0 /media/cdrom0 udf,iso9660 user,noauto,exec,utf8 0 0
/dev/fd0 /media/floppy0 auto rw,user,noauto,exec,utf8 0 0
```

To enable quota, run these commands:

```
touch /quota.user /quota.group
```

```
chmod 600 /quota.*
```

```
mount -o remount /
```

```
quotacheck -avugm
```

```
quotaon -avug
```

13 DNS Server

Run

```
apt-get install bind9
```

For security reasons we want to run BIND chrooted so we have to do the following steps:

```
/etc/init.d/bind9 stop
```

Edit the file /etc/default/bind9 so that the daemon will run as the unprivileged user bind, chrooted to /var/lib/named. Modify the line: OPTIONS="-u bind" so that it reads OPTIONS="-u bind -t

`/var/lib/named":`

`vi /etc/default/bind9`

```
OPTIONS="-u bind -t /var/lib/named"  
# Set RESOLVCONF=no to not run resolvconf  
RESOLVCONF=yes
```

Create the necessary directories under /var/lib:

```
mkdir -p /var/lib/named/etc  
mkdir /var/lib/named/dev  
mkdir -p /var/lib/named/var/cache/bind  
mkdir -p /var/lib/named/var/run/bind/run
```

Then move the config directory from /etc to /var/lib/named/etc:

```
mv /etc/bind /var/lib/named/etc
```

Create a symlink to the new config directory from the old location (to avoid problems when bind gets updated in the future):

```
ln -s /var/lib/named/etc/bind /etc/bind
```

Make null and random devices, and fix permissions of the directories:

```
mknod /var/lib/named/dev/null c 1 3  
mknod /var/lib/named/dev/random c 1 8  
chmod 666 /var/lib/named/dev/null /var/lib/named/dev/random  
chown -R bind:bind /var/lib/named/var/*  
chown -R bind:bind /var/lib/named/etc/bind
```

We need to modify /etc/default/syslogd so that we can still get important messages logged to the system logs. Modify the line: `SYSLOGD=""` so that it reads: `SYSLOGD="-a /var/lib/named/dev/log"`:

`vi /etc/default/syslogd`

```
#  
# Top configuration file for syslogd  
#  
#  
# Full documentation of possible arguments are found in the manpage  
# syslogd(8) .  
#  
#  
# For remote UDP logging use SYSLOGD="-r"  
#  
SYSLOGD="-a /var/lib/named/dev/log"
```

Restart the logging daemon:

```
/etc/init.d/syslogd restart
```

Start up BIND, and check /var/log/syslog for errors:

```
/etc/init.d/bind9 start
```

14 MySQL

In order to install MySQL, we run

```
apt-get install mysql-server mysql-client libmysqlclient15-dev
```

You will be asked to provide a password for the MySQL root user - this password is valid for the user `root@localhost` as well as `root@server1.example.com`, so we don't have to specify a MySQL root password manually later on (as was the case with previous Ubuntu versions):

```
New password for the MySQL "root" user: <-- yourrootsqlpassword
Repeat password for the MySQL "root" user: <-- yourrootsqlpassword
```

We want MySQL to listen on all interfaces, not just localhost, therefore we edit `/etc/mysql/my.cnf` and comment out the line `bind-address = 127.0.0.1`:

```
vi /etc/mysql/my.cnf
```

```
[...]
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address          = 127.0.0.1
[...]
```

Then we restart MySQL:

```
/etc/init.d/mysql restart
```

Now check that networking is enabled. Run

```
netstat -tap | grep mysql
```

The output should look like this:

```
root@server1:~# netstat -tap | grep mysql
tcp    0    0 *:mysql          *:*              LISTEN      5869/mysqld
root@server1:~#
```

15 Postfix With SMTP-AUTH And TLS

In order to install Postfix with SMTP-AUTH and TLS do the following steps:

```
apt-get install postfix libsasl2-2 sasl2-bin libsasl2-modules procmail
```

You will be asked two questions. Answer as follows:

```
General type of mail configuration: <-- Internet Site
System mail name: <-- server1.example.com
```

Then run

```
dpkg-reconfigure postfix
```

Again, you'll be asked some questions:

```
General type of mail configuration: <-- Internet Site
System mail name: <-- server1.example.com
Root and postmaster mail recipient: <-- [blank]
```

Other destinations to accept mail for (blank for none): <-- server1.example.com,
localhost.example.com, localhost.localdomain, localhost
Force synchronous updates on mail queue? <-- No
Local networks: <-- 127.0.0.0/8
Use procmail for local delivery? <-- Yes
Mailbox size limit (bytes): <-- 0
Local address extension character: <-- +
Internet protocols to use: <-- all

Next, do this:

```
postconf -e 'smtpd_sasl_local_domain ='
postconf -e 'smtpd_sasl_auth_enable = yes'
postconf -e 'smtpd_sasl_security_options = noanonymous'
postconf -e 'broken_sasl_auth_clients = yes'
postconf -e 'smtpd_sasl_authenticated_header = yes'
postconf -e 'smtpd_recipient_restrictions =
permit_sasl_authenticated,permit_mynetworks,reject_unauth_destination'
postconf -e 'inet_interfaces = all'
echo 'pwcheck_method: saslauthd' >> /etc/postfix/sasl/smtpd.conf
echo 'mech_list: plain login' >> /etc/postfix/sasl/smtpd.conf
```

Afterwards we create the certificates for TLS:

```
mkdir /etc/postfix/ssl
cd /etc/postfix/ssl/
openssl genrsa -des3 -rand /etc/hosts -out smtpd.key 1024
chmod 600 smtpd.key
openssl req -new -key smtpd.key -out smtpd.csr
openssl x509 -req -days 3650 -in smtpd.csr -signkey smtpd.key -out smtpd.crt
openssl rsa -in smtpd.key -out smtpd.key.unencrypted
mv -f smtpd.key.unencrypted smtpd.key
openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -days 3650
```

Next we configure Postfix for TLS (make sure that you use the correct hostname for myhostname):

```
postconf -e 'myhostname = server1.example.com'
postconf -e 'smtpd_tls_auth_only = no'
postconf -e 'smtp_use_tls = yes'
postconf -e 'smtpd_use_tls = yes'
postconf -e 'smtp_tls_note_starttls_offer = yes'
postconf -e 'smtpd_tls_key_file = /etc/postfix/ssl/smtpd.key'
postconf -e 'smtpd_tls_cert_file = /etc/postfix/ssl/smtpd.crt'
postconf -e 'smtpd_tls_CAfile = /etc/postfix/ssl/cacert.pem'
postconf -e 'smtpd_tls_loglevel = 1'
postconf -e 'smtpd_tls_received_header = yes'
postconf -e 'smtpd_tls_session_cache_timeout = 3600s'
postconf -e 'tls_random_source = dev:/dev/urandom'
```

The file /etc/postfix/main.cf should now look like this:

cat /etc/postfix/main.cf

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version

# Debian specific: Specifying a file name will cause the first
# line of that file to be used as the name. The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# TLS parameters
smtpd_tls_cert_file = /etc/postfix/ssl/smtpd.crt
smtpd_tls_key_file = /etc/postfix/ssl/smtpd.key
smtpd_use_tls = yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

myhostname = server1.example.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = server1.example.com, localhost.example.com, localhost.localdomain,
localhost
relayhost =
mynetworks = 127.0.0.0/8
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
smtpd_sasl_local_domain =
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
smtpd_sasl_authenticated_header = yes
smtpd_recipient_restrictions =
permit_sasl_authenticated,permit_mynetworks,reject_unauth_destination
smtpd_tls_auth_only = no
smtp_use_tls = yes
smtp_tls_note_starttls_offer = yes
smtpd_tls_CAfile = /etc/postfix/ssl/cacert.pem
smtpd_tls_loglevel = 1
smtpd_tls_received_header = yes
smtpd_tls_session_cache_timeout = 3600s
tls_random_source = dev:/dev/urandom
```

Authentication will be done by saslauthd. We have to change a few things to make it work properly. Because Postfix runs chrooted in /var/spool/postfix we have to do the following:

```
mkdir -p /var/spool/postfix/var/run/saslauthd
```

Now we have to edit /etc/default/saslauthd in order to activate saslauthd. Set START to yes and change the line OPTIONS="-c -m /var/run/saslauthd" to OPTIONS="-c -m /var/spool/postfix/var/run/saslauthd -r":

```
vi /etc/default/saslauthd
```

```
#
# Settings for saslauthd daemon
# Please read /usr/share/doc/sasl2-bin/README.Debian for details.
#

# Should saslauthd run automatically on startup? (default: no)
START=yes

# Description of this saslauthd instance. Recommended.
# (suggestion: SASL Authentication Daemon)
DESC="SASL Authentication Daemon"

# Short name of this saslauthd instance. Strongly recommended.
# (suggestion: saslauthd)
NAME="saslauthd"

# Which authentication mechanisms should saslauthd use? (default: pam)
#
# Available options in this Debian package:
# getpwent -- use the getpwent() library function
# kerberos5 -- use Kerberos 5
# pam -- use PAM
# rimap -- use a remote IMAP server
# shadow -- use the local shadow password file
# sasldb -- use the local sasldb database file
# ldap -- use LDAP (configuration is in /etc/saslauthd.conf)
#
# Only one option may be used at a time. See the saslauthd man page
# for more information.
#
# Example: MECHANISMS="pam"
MECHANISMS="pam"

# Additional options for this mechanism. (default: none)
# See the saslauthd man page for information about mech-specific options.
MECH_OPTIONS=""

# How many saslauthd processes should we run? (default: 5)
# A value of 0 will fork a new process for each connection.
THREADS=5

# Other options (default: -c -m /var/run/saslauthd)
# Note: You MUST specify the -m option or saslauthd won't run!
#
# See /usr/share/doc/sasl2-bin/README.Debian for Debian-specific information.
# See the saslauthd man page for general information about these options.
#
# Example for postfix users: "-c -m /var/spool/postfix/var/run/saslauthd"
#OPTIONS="-c -m /var/run/saslauthd"
```

```
OPTIONS="-c -m /var/spool/postfix/var/run/saslauthd -r"
```

Next add the postfix user to the sasl group (this makes sure that Postfix has the permission to access saslauthd):

```
adduser postfix sasl
```

Now restart Postfix and start saslauthd:

```
/etc/init.d/postfix restart  
/etc/init.d/saslauthd start
```

To see if SMTP-AUTH and TLS work properly now run the following command:

```
telnet localhost 25
```

After you have established the connection to your Postfix mail server type

```
ehlo localhost
```

If you see the lines

```
250-STARTTLS
```

and

```
250-AUTH LOGIN PLAIN
```

everything is fine.

The output on my system looks like this:

```
root@server1:/etc/postfix/ssl# telnet localhost 25  
Trying 127.0.0.1...  
Connected to localhost.localdomain.  
Escape character is '^]'.  
220 server1.example.com ESMTP Postfix (Ubuntu)  
ehlo localhost  
250-server1.example.com  
250-PIPELINING  
250-SIZE 10240000  
250-VERFY  
250-ETRN  
250-STARTTLS  
250-AUTH LOGIN PLAIN  
250-AUTH=LOGIN PLAIN  
250-ENHANCEDSTATUSCODES  
250-8BITMIME  
250 DSN  
quit  
221 2.0.0 Bye  
Connection closed by foreign host.  
root@server1:/etc/postfix/ssl#
```

Type

quit

to return to the system's shell.

16 Courier-IMAP/Courier-POP3

Run this to install Courier-IMAP/Courier-IMAP-SSL (for IMAPs on port 993) and Courier-POP3/Courier-POP3-SSL (for POP3s on port 995):

```
apt-get install courier-authdaemon courier-base courier-imap courier-imap-ssl courier-pop courier-pop-ssl courier-ssl gamin libgamin0 libglib2.0-0
```

You will be asked two questions:

```
Create directories for web-based administration? <-- No
SSL certificate required <-- Ok
```

If you do not want to use ISPConfig, configure Postfix to deliver emails to a user's Maildir*:

```
postconf -e 'home_mailbox = Maildir/'
postconf -e 'mailbox_command ='
/etc/init.d/postfix restart
```

***Please note:** You do not have to do this if you intend to use [ISPConfig](#) on your system as ISPConfig does the necessary configuration using procmail recipes. But please go sure to enable Maildir under Management -> Server -> Settings -> EMail in the ISPConfig web interface.

17 Apache/PHP5/Ruby

Now we install Apache:

```
apt-get install apache2 apache2-doc apache2-mpm-prefork apache2-utils libexpat1 ssl-cert
```

Next we install PHP5 and Ruby (both as Apache modules):

```
apt-get install libapache2-mod-php5 libapache2-mod-ruby php5 php5-common php5-curl php5-dev
php5-gd php5-idn php-pear php5-imagick php5-imap php5-mcrypt php5-memcache php5-mhash php5-
ming php5-mysql php5-pspell php5-recode php5-snmp php5-sqlite php5-tidy php5-xmllrpc php5-xsl
```

Next we edit /etc/apache2/mods-available/dir.conf:

```
vi /etc/apache2/mods-available/dir.conf
```

and change the DirectoryIndex line:

```
<IfModule mod_dir.c>

    #DirectoryIndex index.html index.cgi index.pl index.php index.xhtml
index.htm
    DirectoryIndex index.html index.htm index.shtml index.cgi index.php
index.php3 index.pl index.xhtml

</IfModule>
```

Now we have to enable some Apache modules (SSL, rewrite, suexec, and include):

```
a2enmod ssl
a2enmod rewrite
a2enmod suexec
a2enmod include
```

Reload the Apache configuration:

```
/etc/init.d/apache2 force-reload
```

In the next chapter (17.1) we are going to disable PHP (this is necessary only if you want to install ISPConfig on this server). Unlike PHP, Ruby is disabled by default, therefore we don't have to do it.

17.1 Disable PHP Globally

(If you do not plan to install ISPConfig on this server, please skip this section!)

In ISPConfig you will configure PHP on a per-website basis, i.e. you can specify which website can run PHP scripts and which one cannot. This can only work if PHP is disabled globally because otherwise all websites would be able to run PHP scripts, no matter what you specify in ISPConfig.

To disable PHP globally, we edit `/etc/mime.types` and comment out the `application/x-httpd-php` lines:

```
vi /etc/mime.types
```

```
[...]
#application/x-httpd-php                phtml pht php
#application/x-httpd-php-source         phps
#application/x-httpd-php3              php3
#application/x-httpd-php3-preprocessed php3p
#application/x-httpd-php4              php4
[...]
```

Edit `/etc/apache2/mods-enabled/php5.conf` and comment out the following lines:

```
vi /etc/apache2/mods-enabled/php5.conf
```

```
<IfModule mod_php5.c>
  #AddType application/x-httpd-php .php .phtml .php3
  #AddType application/x-httpd-php-source .phps
</IfModule>
```

Then restart Apache:

```
/etc/init.d/apache2 restart
```

18 Proftpd

In order to install Proftpd, run

```
apt-get install proftpd ucf
```

You will be asked a question:

```
Run proftpd: <-- standalone
```

For security reasons add the following lines to `/etc/proftpd/proftpd.conf` (thanks to Reinaldo Carvalho;

more information can be found here: <http://proftpd.org/localsite/Userguide/linked/userguide.html>):

```
vi /etc/proftpd/proftpd.conf
```

```
[...]  
DefaultRoot ~  
IdentLookups off  
ServerIdent on "FTP Server ready."  
[...]
```

ISPConfig expects the configuration to be in `/etc/proftpd.conf` instead of `/etc/proftpd/proftpd.conf`, therefore we create a symlink (you can skip this command if you don't want to install ISPConfig):

```
ln -s /etc/proftpd/proftpd.conf /etc/proftpd.conf
```

Then restart Proftpd:

```
/etc/init.d/proftpd restart
```

19 Webalizer

To install webalizer, just run

```
apt-get install webalizer
```

20 Synchronize the System Clock

It is a good idea to synchronize the system clock with an NTP (**n**etwork **t**ime **p**rotocol) server over the internet. Simply run

```
apt-get install ntp ntpdate
```

and your system time will always be in sync.

21 Install Some Perl Modules Needed By SpamAssassin (Comes With ISPConfig)

Run

```
apt-get install libhtml-parser-perl libdb-file-lock-perl libnet-dns-perl
```

22 ISPConfig

The configuration of the server is now finished, and if you wish you can now install [ISPConfig](#) on it. Please check out the ISPConfig installation manual: http://www.ispconfig.org/manual_installation.htm

22.1 A Note On SuExec

If you want to run CGI scripts under suExec, you should specify `/var/www` as the home directory for

websites created by ISPConfig as Ubuntu's suExec is compiled with /var/www as Doc_Root. Run
/usr/lib/apache2/suexec -V

and the output should look like this:

```
root@server1:~# /usr/lib/apache2/suexec -V
-D AP_DOC_ROOT="/var/www"
-D AP_GID_MIN=100
-D AP_HTTPD_USER="www-data"
-D AP_LOG_EXEC="/var/log/apache2/suexec.log"
-D AP_SAFE_PATH="/usr/local/bin:/usr/bin:/bin"
-D AP_UID_MIN=100
-D AP_USERDIR_SUFFIX="public_html"
root@server1:~#
```

So if you want to use suExec with ISPconfig, don't change the default web root (which is /var/www) if you use expert mode during the ISPConfig installation (in standard mode you can't change the web root anyway so you'll be able to use suExec in any case).

23 Links

- Ubuntu: <http://www.ubuntu.com>
- ISPConfig: <http://www.ispconfig.org>