

Installing WordPress on Ubuntu VPS

A How To

DAILY CUP OF TECH

2008

Authored by: Tim Fehlman

Installing WordPress on Ubuntu VPS

A How To

Step One – Update Server

Edit *sources.list*

```
sudo nano /etc/apt/sources.list
```

Remove all of the # from all commented lines.

Update and Upgrade

```
sudo aptitude -y update
sudo aptitude -y safe-upgrade
sudo aptitude -y full-upgrade
```

Configure Time Zone

```
sudo dpkg-reconfigure tzdata
```

Select the time zone that you are in.

Step Two – Configure *iptables*

Backup present rules

```
iptables-save > /etc/iptables.up.rules
```

Create Filter

Allows all loopback (lo0) traffic and drop all traffic to 127/8 that doesn't use lo0

```
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A INPUT -i ! lo -d 127.0.0.0/8 -j REJECT
```

Accepts all established inbound connections

```
sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Allows all outbound traffic

```
sudo iptables -A OUTPUT -j ACCEPT
```

Allows HTTP and HTTPS connections from anywhere

```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Allows SSH connections (on port 1999)

```
sudo iptables -A INPUT -p tcp -m state --state NEW --dport 1999 -j ACCEPT
```

Allow ping

```
sudo iptables -A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
```

Log iptables denied calls

```
sudo iptables -A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7
```

Reject all other inbound - default deny unless explicitly allowed policy

```
sudo iptables -A INPUT -j DROP
sudo iptables -A FORWARD -j DROP
```

Save Rules

```
sudo iptables-save > /etc/iptables.up.rules
```

Configure Network to Load Rules Automatically

Edit network interface to load rules automatically

```
sudo nano /etc/network/interfaces
```

Add *pre-up iptables-restore < /etc/iptables.up.rules* after *iface lo inet loopback*

Step Three – Install and Configure OpenSSH

```
sudo aptitude -y install openssh-server openssh-client
```

Edit the `sshd_config` file to listen on port 1999 instead of port 22

```
sudo nano /etc/ssh/sshd_config
```

Find *Port 22* and change 22 to 1999 and then save the file.

Restart the SSH server so that the configuration takes hold.

```
sudo /etc/init.d/ssh restart
```

Step Four – Install nginx Web Server

```
sudo aptitude -y install nginx
```

Start the nginx daemon

```
sudo /etc/init.d/nginx start
```

Step Five – Installing MySQL

```
sudo aptitude -y install mysql-server mysql-client libmysqlclient15-dev
libmysql-ruby1.8
```

To not use innodb

```
sudo nano /etc/mysql/my.cnf
```

Then remove comment `#` from the `skip-innodb` line. Save the file.

Step Six - Install PHP5 with fastcgi

```
sudo aptitude -y install php5-common php5-cgi php5-mysql php5-cli php5-gd
```

Create `/etc/init.d/php-fastcgi`

```
sudo nano /etc/init.d/php-fastcgi
```

and add this for its content:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:                php-fastcgi
# Required-Start:          $all
# Required-Stop:           $all
# Default-Start:           2 3 4 5
# Default-Stop:            0 1 6
# Short-Description:       Start and stop php-cgi in external FASTCGI mode
# Description:             Start and stop php-cgi in external FASTCGI mode
### END INIT INFO

# Do NOT "set -e"

PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="php-cgi in external FASTCGI mode"
NAME=php-fastcgi
DAEMON=/usr/bin/php-cgi
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME
PHP_CONFIG_FILE=/etc/php5/cgi/php.ini

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
. /lib/lsb/init-functions

# If the daemon is not enabled, give the user a warning and then exit,
# unless we are stopping the daemon
if [ "$START" != "yes" -a "$1" != "stop" ]; then
    log_warning_msg "To enable $NAME, edit /etc/default/$NAME and set
    START=yes"
    exit 0
fi
```

```

# Process configuration
export PHP_FCGI_CHILDREN PHP_FCGI_MAX_REQUESTS
DAEMON_ARGS="-q -b $FCGI_HOST:$FCGI_PORT -c $PHP_CONFIG_FILE"

do_start()
{
    # Return
    # 0 if daemon has been started
    # 1 if daemon was already running
    # 2 if daemon could not be started
    start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --
    test > /dev/null \
        || return 1
    start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON \
        --background --make-pidfile --chuid $EXEC_AS_USER --startas
    $DAEMON -- \
        $DAEMON_ARGS \
        || return 2
}

do_stop()
{
    # Return
    # 0 if daemon has been stopped
    # 1 if daemon was already stopped
    # 2 if daemon could not be stopped
    # other if a failure occurred
    start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile
    $PIDFILE > /dev/null # --name $DAEMON
    RETVAL="$?"
    [ "$RETVAL" = 2 ] && return 2
    # Wait for children to finish too if this is a daemon that forks
    # and if the daemon is only ever run from this initscript.
    # If the above conditions are not satisfied then add some other code
    # that waits for the process to drop all resources that could be
    # needed by services started subsequently. A last resort is to
    # sleep for some time.
    start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec
    $DAEMON
    [ "$?" = 2 ] && return 2
    # Many daemons don't delete their pidfiles when they exit.
    rm -f $PIDFILE
    return "$RETVAL"
}

case "$1" in
    start)
        [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
        do_start
        case "$?" in
            0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
            2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
    stop)
        [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
        do_stop
        case "$?" in

```

```

        0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
        2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
    esac
    ;;
restart|force-reload)
    log_daemon_msg "Restarting $DESC" "$NAME"
    do_stop
    case "$?" in
        0|1)
            do_start
            case "$?" in
                0) log_end_msg 0 ;;
                1) log_end_msg 1 ;; # Old process is still running
                *) log_end_msg 1 ;; # Failed to start
            esac
        ;;
        *)
            # Failed to stop
            log_end_msg 1
        ;;
    esac
    ;;
*)
    echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload}" >&2
    exit 3
    ;;
esac

```

Make `/etc/init.d/php-fastcgi` executable:

```
sudo chmod 755 /etc/init.d/php-fastcgi
```

Create `/etc/default/php-fastcgi`

```
sudo nano /etc/default/php-fastcgi
```

and add this for its content:

```

START=yes

# Which user runs PHP? (default: www-data)

EXEC_AS_USER=www-data

# Host and TCP port for FASTCGI-Listener (default: localhost:9000)

FCGI_HOST=localhost
FCGI_PORT=10005

# Environment variables, which are processed by PHP

PHP_FCGI_CHILDREN=4
PHP_FCGI_MAX_REQUESTS=1000

```

Create `/etc/nginx/fcgi.conf`

```
sudo nano /etc/nginx/fcgi.conf
```

and add this for its content:

```
fastcgi_param QUERY_STRING $query_string;
fastcgi_param REQUEST_METHOD $request_method;
fastcgi_param CONTENT_TYPE $content_type;
fastcgi_param CONTENT_LENGTH $content_length;

fastcgi_param SCRIPT_NAME $fastcgi_script_name;
fastcgi_param REQUEST_URI $request_uri;
fastcgi_param DOCUMENT_URI $document_uri;
fastcgi_param DOCUMENT_ROOT $document_root;
fastcgi_param SERVER_PROTOCOL $server_protocol;

fastcgi_param GATEWAY_INTERFACE CGI/1.1;
fastcgi_param SERVER_SOFTWARE nginx/$nginx_version;

fastcgi_param REMOTE_ADDR $remote_addr;
fastcgi_param REMOTE_PORT $remote_port;
fastcgi_param SERVER_ADDR $server_addr;
fastcgi_param SERVER_PORT $server_port;
fastcgi_param SERVER_NAME $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param REDIRECT_STATUS 200;
```

Edit `/etc/php5/cgi/php.ini` to allow for each script to use up to 32 MB of RAM.

```
sudo nano /etc/php5/cgi/php.ini
```

Look for `memory_limit = 16M` and change it to `32M`. Save the file.

Start the process:

```
sudo /etc/init.d/php-fastcgi start
```

Configure the process to start on reboot:

```
sudo update-rc.d php-fastcgi defaults
```

Restart nginx.

```
sudo /etc/init.d/nginx stop
sudo /etc/init.d/nginx start
```

Step Seven – Create the Website

Create Directory Structure

```
mkdir -p
/home/<username>/public_html/<domain.name>/{public,private,log,backup}
```

where `<username>` is your username in the VPS and `<domain.name>` is the name of the domain that you are creating.

Create Default Index Page

In `/home/<username>/public_html/<domain.name>/public/`, create an `index.php` file as a placeholder. I generally create the standard `phpinfo` page:

```
<?php
phpinfo();
?>
```

Create the vhost File

Open `/etc/nginx/sites-available/<domain.name>` in a text editor as root:

```
sudo nano /etc/nginx/sites-available/<domain.name>
```

where `<domain.name>` is the website that you are creating.

Add the following as the content, replacing `<domain.name>` and `<username>` accordingly:

```
server {

    listen 80;
    server_name <domain.name>;
    #rewrite ^/(.*) http://www.<domain.name> permanent;

    access_log /home/<username>/public_html/<domain.name>/log/access.log;
    error_log /home/<username>/public_html/<domain.name>/log/error.log;

    location / {

        root /home/<username>/public_html/<domain.name>/public/;
        index index.php;
        # wordpress fancy rewrites
        if (-f $request_filename) {
            break;
        }
        if (-d $request_filename) {
            break;
        }
        rewrite ^(.+)$ /index.php?q=$1 last;

    }

    location ~ .*\.php[345]?$ {
        include /etc/nginx/fastcgi.conf;
        fastcgi_pass 127.0.0.1:10005;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME
/home/<username>/public_html/<domain.name>/public$fastcgi_script_name;
    }

}

server {

    listen 80;
    server_name www.<domain.name>;
```

```

access_log /home/<username>/public_html/<domain.name>/log/access.log;
error_log /home/<username>/public_html/<domain.name>/log/error.log;

location / {

    root /home/<username>/public_html/<domain.name>/public/;
    index index.php;
    # wordpress fancy rewrites
    if (-f $request_filename) {
        break;
    }
    if (-d $request_filename) {
        break;
    }
    rewrite ^(.+)$ /index.php?q=$1 last;

}

location ~ .*\.php[345]?$ {
    include /etc/nginx/fastcgi.conf;
    fastcgi_pass 127.0.0.1:10005;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME
/home/<username>/public_html/<domain.name>/public$fastcgi_script_name;
}

}

```

Enable the Website

```

sudo ln -s /etc/nginx/sites-available/<domain.name> /etc/nginx/sites-
enabled/<domain.name>

```

Restart nginx

```

sudo /etc/init.d/nginx stop
sudo /etc/init.d/nginx start

```

Step Eight - Install and Configure the Mail Server

Installation

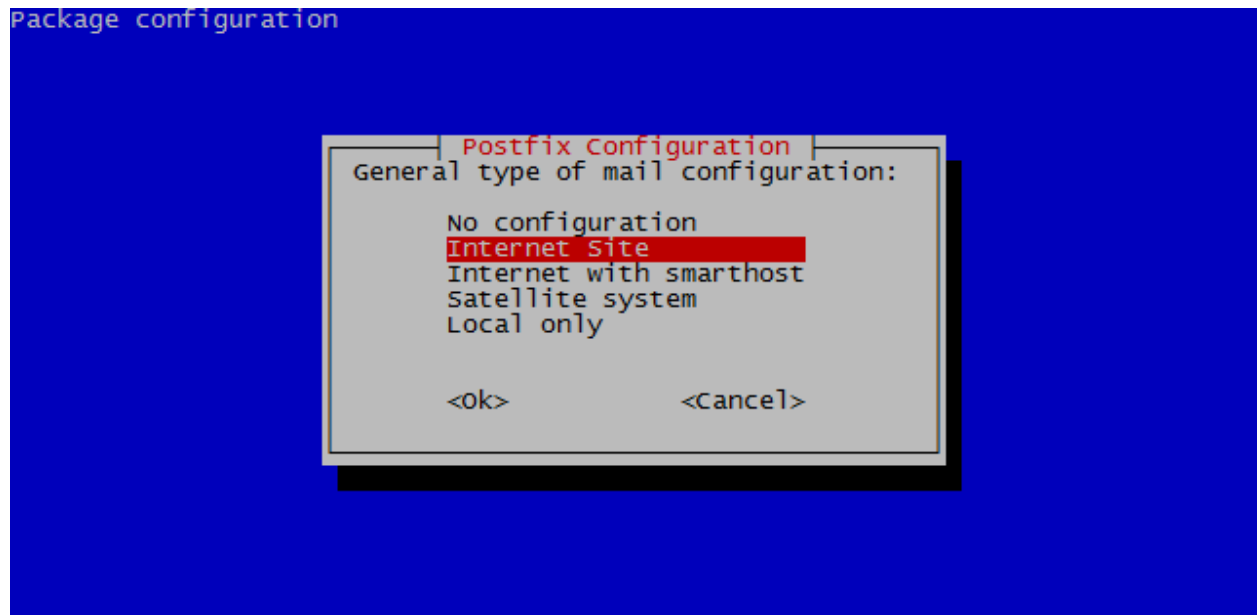
Install Postfix mail server and telnet for testing:

```

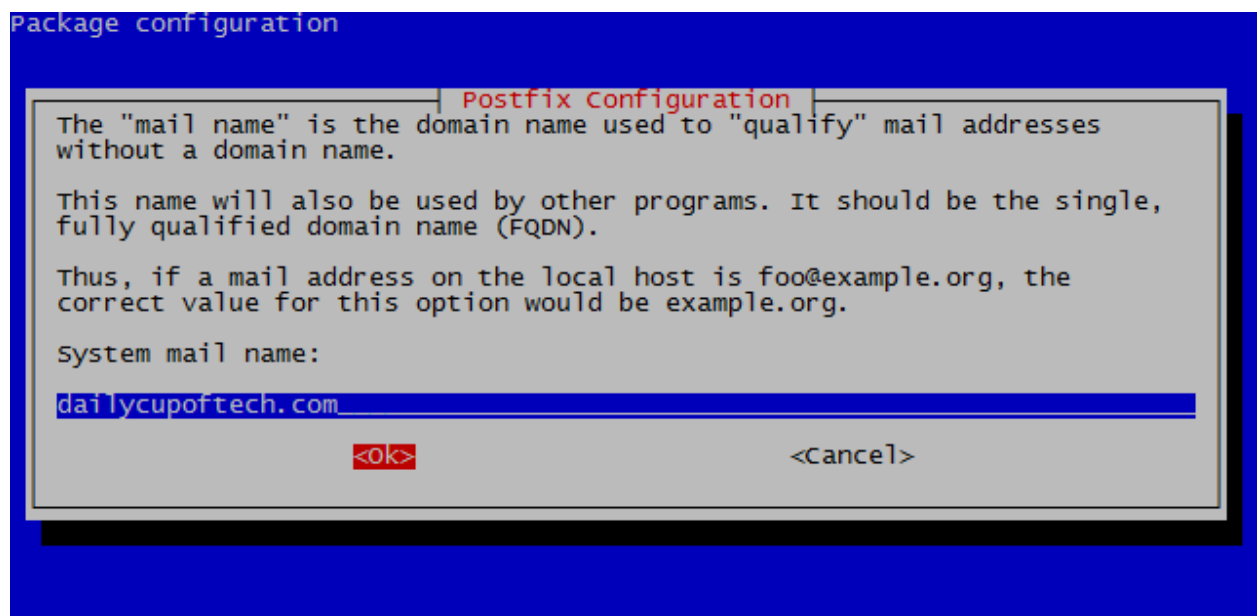
sudo aptitude install postfix telnet

```

When prompted for the type of mail configuration, select *Internet Site*:



Enter the domain name that you will be hosting:



Test Installation

Attempt to connect to the mail server using telnet by entering the following command:

```
telnet localhost 25
```

The response that you get back should be something similar to this:

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 VPS.localdomain ESMTP Postfix (Ubuntu)
```

Step Nine – Install WordPress via Subversion

Install Subversion

Using Subversion is a much easier way to install WordPress. Install Subversion with this command:

```
sudo aptitude install subversion
```

Install WordPress

Move your working directory to the root of the directory that you want to install WordPress in:

```
cd /home/<username>/public_html/<domain.name>/public
```

Install the latest stable version of WordPress:

```
svn co http://svn.automattic.com/wordpress/tags/2.6 .
```

Step Ten – Importing the Old WordPress Website Files

Extract files from the archive to a folder on your workstation. Then copy the contents of the *wordpress* folder to */home/<username>/public_html/<domain.name>/public*. One of the easiest ways of doing this is to use WinSCP.

Step Eleven – Import Data to SQL Server

Copy the *wordpress.sql* file to *~* using WinSCP.

Log on to the MySQL server.

```
mysql -u root -p
```

Enter your password when prompted.

Create the *wordpress* database.

```
create database wordpress;
```

Connect to the *wordpress* database.

```
connect wordpress;
```

Import the *wordpress.sql* file into the *wordpress* database.

```
source wordpress.sql;
```

Be patient as this may take some time, depending on the amount of data you have. Once the import is complete, exit out from the MySQL client.

```
exit
```

Step Twelve – Editing wp_config.php

With new information, it is important to update the *wp_config.php* file.

```
nano /home/<username>/public_html/<domain.name>/public/wp-config.php
```

Edit the file with all of the appropriate information. Save the file.