

[The Ultimate Open Source Web Server Installation](#) [Part 1](#)

September 30th, 2008



In this tutorial, I'm going to walk you through the process of setting up your own web server using the best open source software available. Whether you're hosting the server yourself or configuring your own "slice" at [Slicehost](#), knowledge of how your web server functions is vital and there's no better way to learn about your web server than installing and configuring it yourself. In this tutorial, we'll be building a web server using [Ubuntu 8.04.1](#) as our operating system. We'll spend Part I installing Ubuntu and setting up SSH for remote login as well as configuring a simple firewall. In Part II, we'll install our web server technologies and server-side programming languages along with other little tools that will come in handy. Some of these include [NGINX](#), a powerful and simple web server, [Ruby](#), [PHP](#), and [MySQL](#), [Subversion](#), [SQLite3](#), and [PHPMyAdmin](#). In Part III we'll go into detail about installing popular frameworks and content management systems like [Ruby on Rails](#), [Wordpress](#), and [Joomla!](#).

For this installation, I'm using the latest release of the Ubuntu Server v8.04.1. I downloaded the full disc iso from the Ubuntu site and burned it to a cd. Go ahead and pop that into the server's cd-rom and let's get started!

I'm going to walk you through the process with the settings that I chose for my server setup. I am not a Linux guru or a web server guru. I don't have a 100% understanding of how everything works but I've gathered enough info on the web to figure out how I need to customize the installation according to my needs. This post is compiled of steps and configurations that I've found on the internet. I wanted to have a consolidated instruction manual so I'm writing this tutorial to help anyone who's in the same boat as me.

DISCLAIMER: Follow these instructions at your own risk. Though these steps have worked for me, I can not take responsibility for anything that goes wrong on your end. That takes care of the legal issue - let's get started ! 😊

When the installation process starts choose English



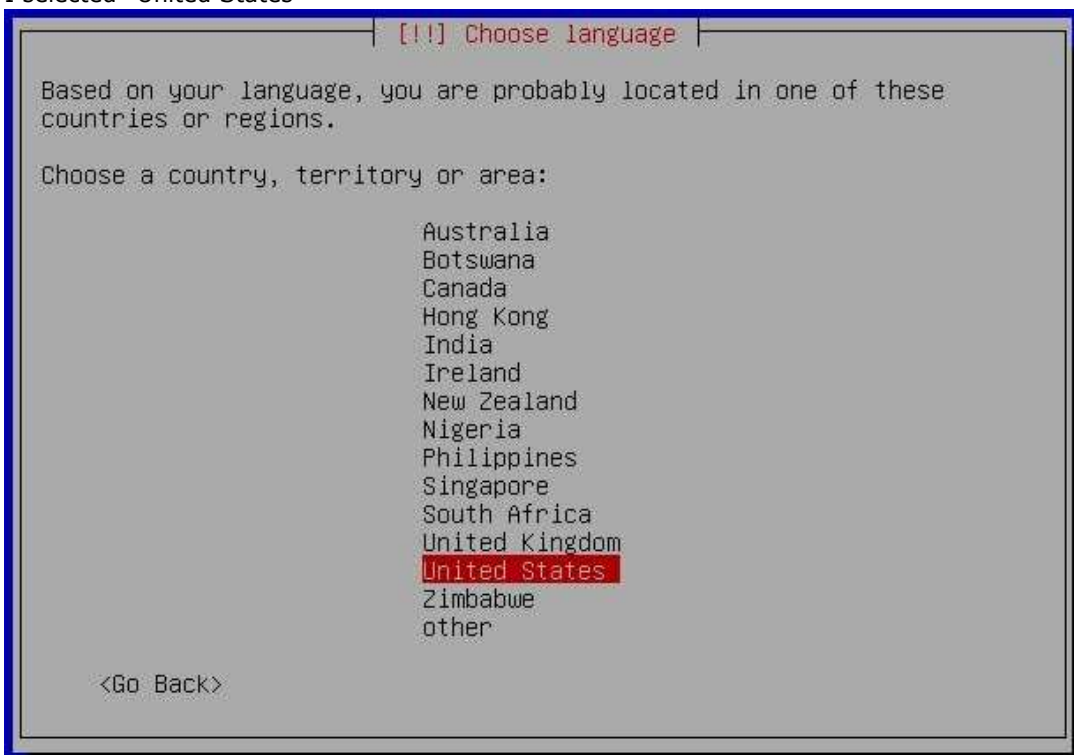
Select "Install Ubuntu Server"



Select English (if you speak english - which I gather you do if you're reading this)



I selected "United States"



Choose your keyboard layout (I chose to skip this step and I chose "USA" in the following windows)

[!] Ubuntu installer main menu

You can try to have your keyboard layout detected by pressing a series of keys. If you do not want to do this, you will be able to select your keyboard layout from a list.

Detect keyboard layout?

<Go Back>

<Yes>

<No>

[!] Ubuntu installer main menu

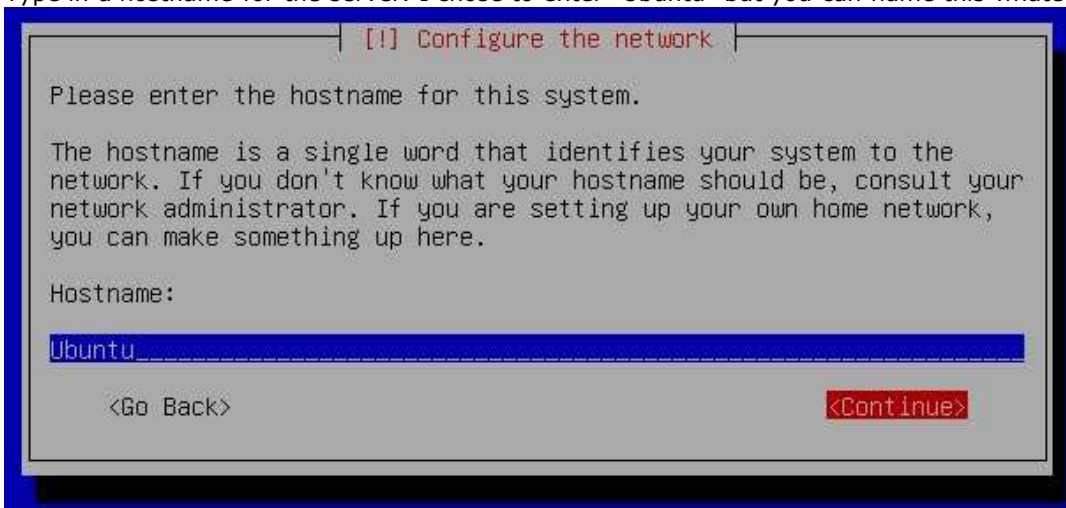
The origin of the keyboard:

- Norway
- Pakistan
- Poland
- Portugal
- Romania
- Russia
- Serbia
- Slovakia
- Slovenia
- South Africa
- Spain
- Sri Lanka
- Sweden
- Switzerland
- Syria
- Tajikistan
- Thailand
- Turkey
- USA

<Go Back>



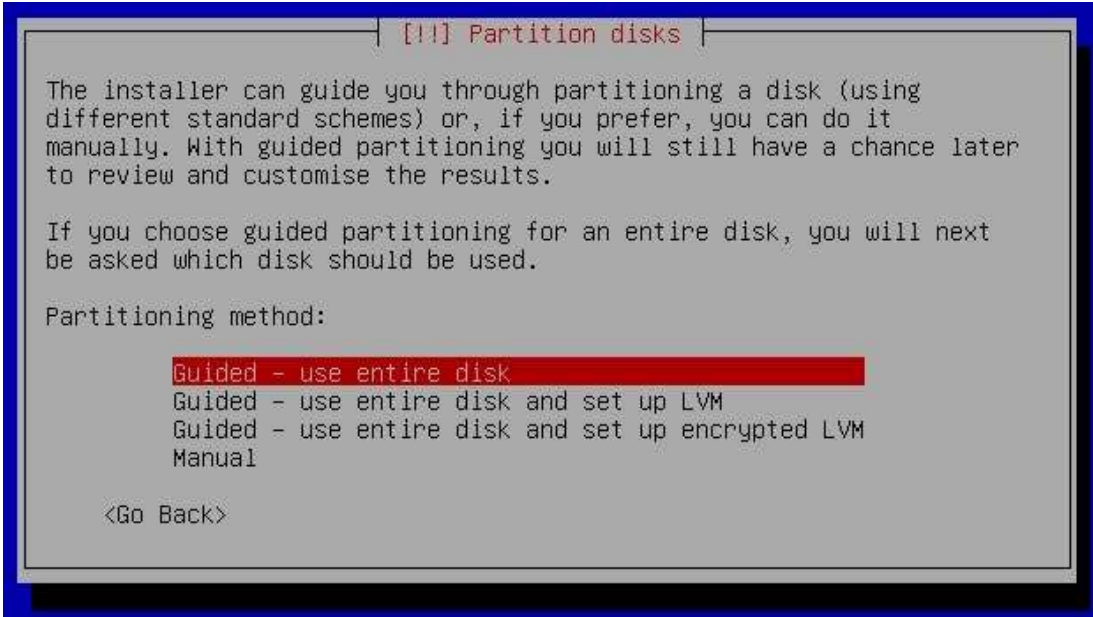
Type in a hostname for the server. I chose to enter "Ubuntu" but you can name this whatever you'd like.



Select your time zone.



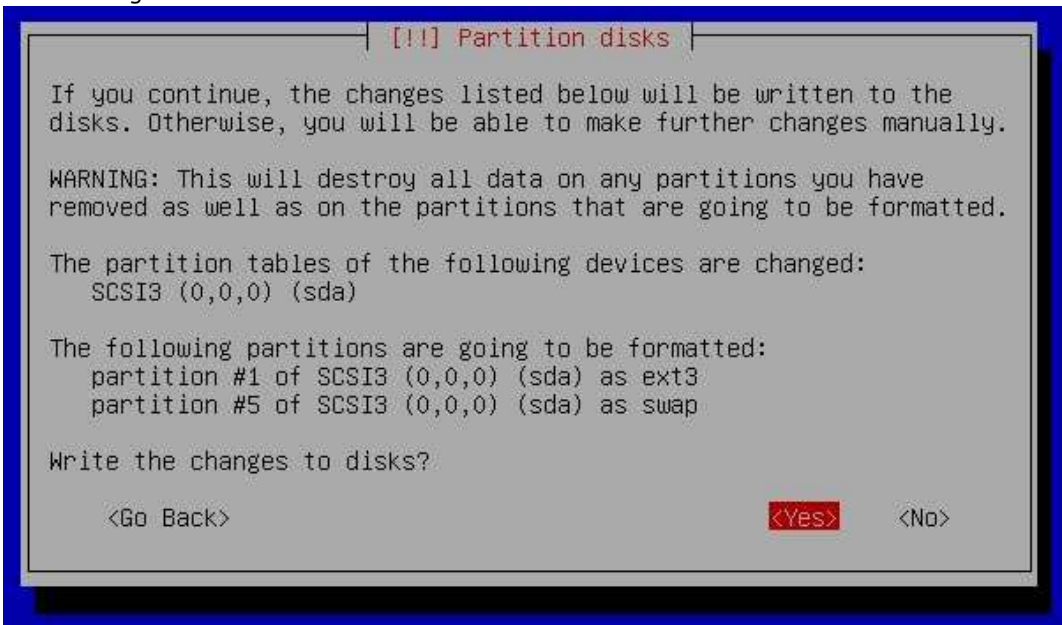
Disk Partition: I chose to "Guided - use entire disk" because I want to use the entire disk.



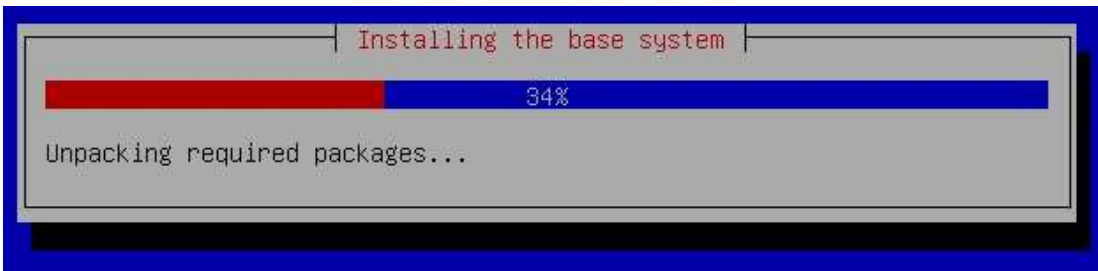
Hit enter to confirm.



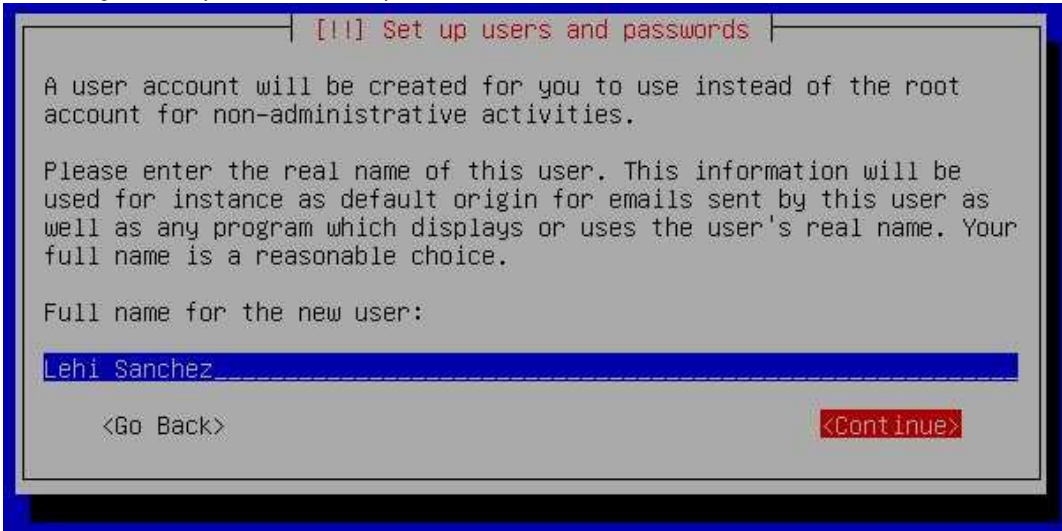
Write changes to disk: Select "Yes" to confirm.



Depending on your system this process could take a few minutes

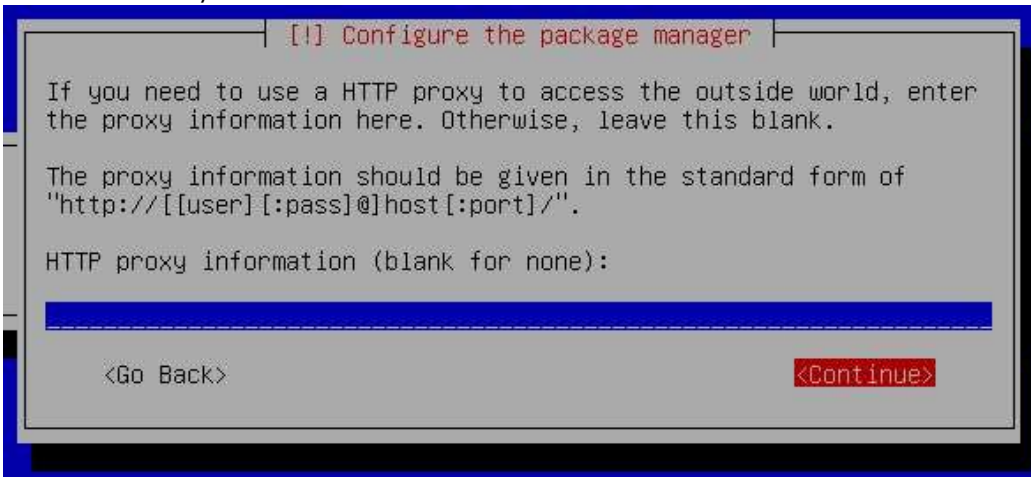


Now we just setup our user and passwords.

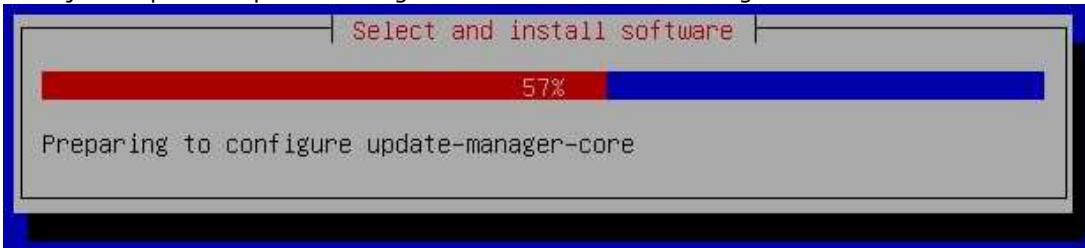




Leave HTTP Proxy blank.



We'll just skip the step of installing software since we'll be doing this ourselves later.



Once the installation is complete, remove the disc and hit continue. The system will restart automatically and you will be brought to the command prompt.



When the installation is complete take the cd rom out of your drive and hit continue. The computer will restart automatically. You will then be brought to the command prompt.

Updating The Server

Before we start installing anything, let's make sure our system is up to date. Use the following to bring your system up to date. (some of these settings are country specific)

```
sudo aptitude update
sudo locale-gen en_US.UTF-8
sudo /usr/sbin/update-locale LANG=en_US.UTF-8
sudo aptitude safe-upgrade
sudo aptitude full-upgrade
sudo aptitude install build-essential
```

SSH

In most cases, you'll be accessing the server from a different computer. Let's install ssh so we can login remotely and securely.

```
sudo apt-get install ssh
```

Once that is complete, we'll need to know the server's IP address. If you don't know the address, simply type the following.

```
ifconfig
```

Look for the "inet addr:"

```
administrator@Ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:00:00:00:00
          inet addr:172.0.0.1  Bcast:172.0.0.255  Mask:255.255.255.255
          inet6 addr: 1234::123:1234:1234:1234/12  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1435  Metric:2
          RX packets:45000 errors:0 dropped:0 overruns:0 frame:0
          TX packets:25002 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1010
          RX bytes:47228865 (45.0 MB)  TX bytes:1441993 (1.2 MB)
```

This is the IP address (your's might be different). Write this down, we'll need it to login in a little bit.

```
inet addr:172.0.0.1
```

Login

On another machine, open up terminal and type the following.

```
ssh your-username@server-ip-address
```

You might get a message that says

```
The authenticity of host '172.0.0.1 (172.0.0.1)' can't be established.
RSA key fingerprint is **:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*.
Are you sure you want to continue connecting (yes/no)?
```

Type 'yes'. You'll get a response similar to the following.

Warning: Permanently added '172.0.0.1' (RSA) to the list of known hosts.

Enter the user's password. You should now be presented with a new name at the prompt like this.

```
your-username@server-name:~$
```

Now let's copy the ssh key over to the server. On your LOCAL computer type:

```
scp ~/.ssh/id_rsa.pub administrator@172.0.0.01:/home/administrator/
```

Now on your SERVER type:

```
mkdir /home/your-username/.ssh
mv /home/administrator/id_rsa.pub /home/administrator/.ssh/authorized_keys
chown -R administrator:administrator /home/administrator/.ssh
chmod 700 /home/administrator/.ssh
chmod 600 /home/administrator/.ssh/authorized_keys
```

Now we'll open up the ssh file and tweak some settings.

```
sudo pico /etc/ssh/sshd_config
```

look for the following items and change them so they each look like this

```
Port 8912 # <--- you can change this to what ever port number you'd like
Protocol 2
PermitRootLogin no
PasswordAuthentication yes # <-- set to no if you are just using one computer to connect to the
server
X11Forwarding no
UsePAM no
UseDNS no
AllowUsers administrator # <-- your username
```

Save the file by hitting ctrl-x and 'y' and enter.

Security

Again, before we do anything else, let's lockdown our server with a simple but effective firewall. Please note: that this is just a simple firewall - be responsible enough to strengthen it according to your needs.

```
sudo -s
iptables-save > /etc/iptables.up.rules
nano /etc/iptables.test.rules
```

Copy and paste the following into the new file

```
*filter
# Allows all loopback (lo0) traffic and drop all traffic to 127/8 that doesn't use lo0
-A INPUT -i lo -j ACCEPT
-A INPUT -i ! lo -d 127.0.0.0/8 -j REJECT
..
```

```

# Accepts all established inbound connections
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allows all outbound traffic
# You can modify this to only allow certain traffic
-A OUTPUT -j ACCEPT

# Allows HTTP and HTTPS connections from anywhere (the normal ports for websites)
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT
-A INPUT -p tcp --dport 21 -j ACCEPT

# Allows SSH connections
#
# THE -dport NUMBER IS THE SAME ONE YOU SET UP IN THE SSHD_CONFIG FILE
#
# CHANGE THE FOLLOWING PORT NUMBER TO THE ONE YOU CHOSE
-A INPUT -p tcp -m state --state NEW --dport 8912 -j ACCEPT

# Allow ping
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT

# log iptables denied calls
-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
-A INPUT -j REJECT
-A FORWARD -j REJECT

COMMIT

```

Then

```

iptables-restore < /etc/iptables.test.rules
iptables-save > /etc/iptables.up.rules
nano /etc/network/interfaces

```

Add a line just after 'iface lo inet loopback':

```

...
auto lo
iface lo inet loopback
pre-up iptables-restore < /etc/iptables.up.rules

# The primary network interface
...

```

Save the file and then type:

```
/etc/init.d/ssh reload
```

Open a new tab in the terminal and try to connect to your server using this pattern:

```
ssh -p 8912 your-username@server-address
```

If you can't login, something went wrong. I can't provide too much information now for fixing the problem other than repeating the steps and googling it.

Review

Let's review of what we did just now. We:

- Installed Ubuntu Server
- Updated the system
- Installed and Configured SSH
- Setup a small and workable firewall

In the next article, we'll finally install all of the software I've been promising!

* If there are any mistakes in what you've read so far - please leave a comment and I'll see if I can update the post for future readers. Thanks!!