



## **The CitizenWeb Guides**

- Getting Started with Linux
- Setting Up Your Personal Server  
and more

**Version 1.0**  
January 2013

## Table of Contents

1.1. “What is Free Software, and Why Do I Give A Damn?” The Case for Making The Switch.....	3
1.2. “What's Wrong With Google?” Security, Safety and Rights on the Internet.....	7
1.3. A Manifesto for a Decentralized Web.....	10
2.1. Choosing a Distribution.....	13
2.2. Installing Ubuntu.....	22
2.3. Getting Used to Ubuntu.....	31
2.4. Securing Web, Email and Chat Applications.....	41
2.5. APPENDIX: Popular Applications.....	57
3.1. Why a Personal Server?.....	64
3.2. Before You Begin: Options, Configuration and Hardware.....	68
3.3. Assemble Your PC.....	74
3.4. Installing Ubuntu Server.....	75
3.5. Getting In: Using SSH and VNC.....	84
3.6. Home Networking: DHCP, DNS and NAT.....	89
3.7. Host Your Email: Setting Up Postfix and Dovecot.....	99
3.8. Host a Website with Apache and PHP.....	106
3.9. Your Own “Cloud”: Files, Calendar and Contacts.....	115
3.10. Security: Firewalling and Threat Detection.....	123
3.11. Managing and Streaming Your Media.....	128
3.12. APPENDIX: Guide to Virtual Machines.....	133
3.13. APPENDIX: Guide to FreeNAS.....	150
4.1 Backup and Encrypt Your Data.....	159

## ***The CitizenWeb Guides - Introduction***

### **1.1. "What is Free Software, and Why Do I Give A Damn?" The Case for Making The Switch**

---

The traditional definition of "free software" has varied slightly over the years, and has multiple meanings depending on which member of the community one is talking to. Yes, oftentimes free software can mean software that is "free as in beer," i.e. receiving a product for free and not needing to pay in order to use it. This is definitely a good aspect to most free software, however the more important definition is the one that is more widely intended when one speaks of "free software." Free as in "libre," that is, software that opens its source code to public viewing and adaptation. This is contrary to closed-source software like the Windows or OS X operating systems, which do not release their source code and therefore cannot be modified or independently verified by members of the general public.

Now, most partisans of free software advocate for its use based on a quasi-moral or altruistic argument. Free software should be used because it puts users in control of their own computers, because it doesn't lock users into so-called "walled gardens" that force them to choose certain options, et cetera. Never before have we been confronted with such a narrowing technological environment -- Apple wants to lock its users into using iDevices, only getting software from its closely-watched App Stores, and locked out of any sort of meaningful configuration of their own computers. Microsoft and Google are not too far behind Apple's lead in this regard. Therefore free software represents a clear alternative to these "un-free" systems of control. This approach to arguing for free software is all well and good, but it doesn't attack at the central problem with free software: its perception as a hobbyist operating system, unreliable and only for advanced use. You can give all the moral arguments in the world, but as we have seen throughout history, these rarely make deep imprints in human behaviour.

For the unconvinced, here is the primary reason why you should make the switch to Linux and free software: **because in nearly every case, it provides you with the best computing environment available, with the most features and most customizable and dynamic interface on the market today.**

To all the Mac fanboys out there, I'm sorry for making you spit out your tea and crumpets. But it's true, and I'll explain why.

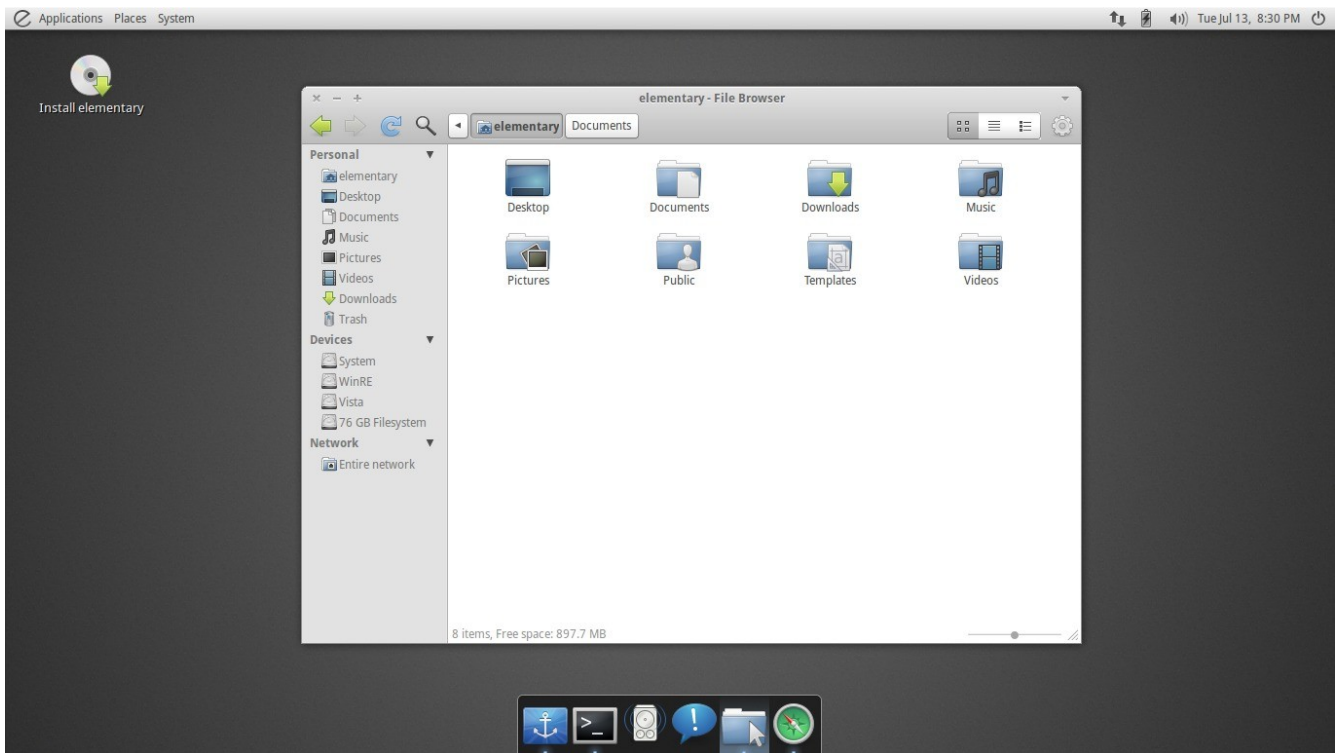
As referenced earlier, free software opens its source code to the general public so that anyone can verify it or modify the program to their liking. It should be noted that a significant proportion of general users would never feel the need to do something like this. Just the same as how people wouldn't want to use Linux based on a conception of it as a hobbyists' operating system. The benefits are not just isolated to the end user, though: when using free software, you get the assurance that the software has most likely been vetted by prior users and developers, to grant it greater credibility. Free software that has been downloaded direct from the developer or a secure repository is much less likely to contain back doors, malicious code, spyware or other nasties prevalent in proprietary software.

In addition to this, open software has a much higher degree of usability because of its openness. Say there is a functionality in a piece of software that just doesn't make sense to you, and you wish you could either turn it off or use another program that works in a different way. In Windows, you are much more likely to be stuck with Microsoft's whims, locking you into a particular software suite or way of doing things. If not, then you may have to pay in order to get full access rights to a new application. With free software, we don't have to worry about any of that. If you have some programming skill, you can easily poke around the source code and adjust the functionality of your favourite programs, and you are fully within your right to do so. Or, better yet, you are free to surf through the repositories or online databases like Github in order to find a suitable alternative. This more democratized software development process breathes healthy competition into the software market, which can only benefit the end user.

Now you may say "Linux might be great, but it simply isn't an operating system for daily use!" To which I would respond: today's Linux has advanced dramatically from what it was fifteen, ten, even five years ago. It isn't like that old RedHat box you played around with back in the mid-90s. Interfaces for most major distributions like Ubuntu and Linux Mint have been polished considerably well. Each major distribution has its preferred display environment, and each one looks and performs just as well as their proprietary competitors.

### 1.1. “What is Free Software, and Why Do I Give A Damn?” The Case for Making The Switch

Take a look at elementaryOS, for example, which tries to emulate Mac OS X's signature visual style:



Versus Mac OS X:



Modern Linux distributions like elementaryOS put a high priority on sleek and functional user interfaces. Fedora Linux comes with GNOME3. Linux Mint has Cinnamon. Ubuntu has Unity which, while it is often maligned by many in the free software community, has been making serious improvements in recent years. And there are many other options to choose from, all of which give you easy and intuitive interfaces without the need to muddle through the Terminal or obscure command switches. Take your pick -- you don't have to settle for the godawful mess that is Windows 8, or the suffocating money sink that is Mac OS X.

Free software isn't just limited to operating systems, either. Got a bone to pick with Microsoft Office? Try LibreOffice. Don't want to give Google Analytics all of your site visitors' data? Check out Piwik. Addicted to iPhoto but don't want to pay a king's ransom for a new MacBook? Take a look at Shotwell. Why would you pay Apple hundreds of dollars to use Time Machine, iTunes, or iCloud when all of these systems are freely available on Linux, and are by most accounts even better? Why would you pay Microsoft to lock you into their ridiculous Windows 8 Metro interface, when you can have a computer that works exactly how you want it, with better performance and (usually) better stability? Even if there is a program that only comes on Windows that you absolutely *must* have, these can be run via virtual machine systems like VirtualBox, making it easier than ever to have the best of both worlds.

For nearly every proprietary software platform in use these days, there is a tried and true open source alternative. Some of them are more advanced than others, but for general-purpose daily computing, Linux and free software provide the most advanced and customizable user experience available -- one that is also increasingly stable and hardware-friendly.

## 1.2. "What's Wrong With Google?" Security, Safety and Rights on the Internet

---

I should begin this section with saying that there is nothing *\*inherently\** wrong with using and improving your life with an internet services platform like Google. Nor is there anything inherently wrong with using a Windows operating system. Or Apple products, for that matter. Billions of people around the world use these systems everyday without negative consequence. Their advancement has provided untold ability to learn and improve life for nearly everyone on the planet, that much is certain. The problem with services like Google lie in their newfound ubiquity, as well as their ability to store vast amounts of data on us - including details as personal as our *\*physical location\** - with little to no external oversight. And our continued use of these services enable and affirm such moves, providing these services with the justification they need to continue on their privacy onslaught. Google's success in propagating itself to every corner of our lives - with our full acquiescence - is the reason we should be so determined to resist it.

The questions here are simple: What fundamental responsibility do we have to our own information? What are our personal details, our meetings and writing, our entire lives that are now exceedingly being stored and lived on the internet, really worth to us? What rights do we really have when we use publicly-available services with privacy policies that are dozens of pages long? This is something that only each individual can decide for themselves. But these questions are only becoming more pertinent. As an incredible amount of our lives these days is lived on the Internet, it merits a very serious and sober look at just what we own and who we give it to for "safekeeping."

---

When there is no external oversight over an organization that safeguards our data, you must trust that organization to always act in your best interests. Once upon a time, Google's slogan was "do no evil." Those days are, of course, [now long gone](#). Google's quest for power and consolidation of the internet services market has reached a fever pitch. This closing of the online ecosystem has given it (and, by extension, its advertisers) unprecedented and centralized access to our personal data.

The centralization of data on large platforms such as Google provides new and substantial improvements to the ease-of-use and the ease-of-access we experience in using our data. Unfortunately there is a corresponding improvement in corporate and governmental access

to the same data. Not only do these entities have to cut a considerable amount of time and red-tape out of their information gathering operations by only having to deal with one platform, they also win by being able to standardize their approaches against one uniform set of rules and policies for this platform.

While Google has done a notable job in providing transparency when it comes to "official" government takedown requests on its various services, one can see that the amount of them are growing each and every year. Not all of them are granted (thankfully) but this is only due to Google's insistence. When the financial incentive to resist no longer swings their way, however, one will find that even the most well-intentioned company will change their tune remarkably quick. When your last line of defence for your data is trusting in a corporation, which has its own prerogatives and incentives, this defence is a weak one indeed.

Most believe that if they do not break the law online, they will not be targeted by governments. The age-old slogan "If you've done nothing wrong, you have nothing to hide," has lost any merit it may have ever had. In these days of wireless surveillance, we know that any individual can be caught up in the fray. From the US and UK's monitoring of Occupy activists to the Obama administration's breathless expansion of state surveillance powers, governments around the world have raced to prove that, even if you stand up for a cause you believe in, peacefully and well within your "rights," you *\*will\** be targeted. Even if you are a simple bystander, your personal data can be rifled through with impunity. Personal information of innocent people is constantly being [vacuumed up and sifted through](#) by the national security establishment. Quite simply, "rights" on the Internet's public services do not exist.

Even if you've "done nothing wrong," AND you don't care about government spooks looking through your daily calendar, it's even more absurd that companies like Facebook and Google are gathering huge volumes of advertising data on us without most people knowing. This data can be used to create intricate profiles of our daily lives, giving companies much more information than we may even know about our own selves. An Austrian law student currently pursuing Facebook in court [found that the company had more than 1,000 pages of data on him](#). This would not only include his favourite movies and drunk self-portraits: intimate details of his browsing history (on Facebook or elsewhere), and advertising profiles created based on the things he's viewed, liked and subscribed to. Facebook created his very own consumer image, and this data gets sold to advertising groups around the world. Unaccountable corporations can then trade in your personal data for them to enrich themselves at your expense. This brings up an even more confounding fundamental question: why do we let companies like Facebook monetize our universe like this with impunity? Is it right that companies get to sell our intimate details without our knowledge for their astronomical profit?



When it comes to security and data rights online, things are only moving in one direction. And that is towards more control for large corporations and governments, and less control for individual users. In the best of cases, this means our private and intimate data being used to enrich morally unscrupulous corporations. In the worst, it means surveillance, monitoring and snooping for those who express an opinion the government might not endorse.

These serious concerns for safety and privacy can only be countered with a cohesive strategy for personal data liberation and independence. This guide aims to provide detailed instructions for the common user to enact just such a strategy for themselves, while keeping every bit of the comfort and ease-of-use that large internet services like Google can provide.

*"Value your freedom or you will lose it, teaches history. 'Don't bother us with politics', respond those who don't want to learn."* - **Richard M. Stallman**

## Further Reading

- ["Google Transparency Report Shows Rising Trend of Government Surveillance" - Electronic Freedom Foundation \(EFF\)](#)
- ["When Will our Email Betray Us? An Email Privacy Primer in Light of the Petraeus Saga" - Electronic Freedom Foundation \(EFF\)](#)
- ["Activist Requests Her FBI File, Learns What Color Hat She Was Wearing When She Went to See 'Lord of the Rings'" - The Stranger](#)

## 1.3. A Manifesto for a Decentralized Web

---

### We BELIEVE...

- **that an individual's control of their own selves is paramount.** As our society advances, and as the slow merger of technology with our natural thoughts and actions progresses, the individual must be given the means assert control over their own virtual selves.
- **that the best form of assurance is personal control.** The best way to keep the security of one's data is to keep it within one's own reach.
- **that the fair right to free and open communication cannot be abridged.** Never before have we lived in an era where governments and corporations position themselves as such titanic gatekeepers of communication. These gatekeepers cannot be humanity's intermediaries.
- **that the self must be strengthened so that society might flourish.** We do not speak of resistance to control as vulgar individualists. Cohesion in society and benefit to all, regardless of race, creed, class or other hierarchy, cannot be attained without free and unthrottled communication.
- **that the amount of data gathered in one space is directly proportional to the amount of interest governments and corporations take to controlling it.** Information is the new gold, whether it is sensitive personal data or minable marketing statistics. Anywhere it is amassed, there will be forces attempting to control it.
- **that the amount of data gathered in one space is directly proportional to the EASE with which governments and corporations can control it.** One warrant is easier to get than one hundred, and one financially-interested company is easier to intimidate than one hundred individual users. Furthermore, the ease with which governments can directly intercept communications grows when they can connect themselves directly to these platforms.
- **that open development is the most reliable way to assure something's working order.** As we become more and more dependent on technology, it becomes easier to ignore its inner workings. Only technology developed according to "open source" principles can be verified to function in a safe and secure manner.
- **that the ability to keep communication or data private from others is a right.** Whether its by an assured method like encryption, or simply by only publishing in a selective way, users who have not infringed on the rights of another should expect a

default state of privacy.

- **that freedom of expression comes from the assertion of natural right, and is not given freely.** Change will not occur unless it is demanded and fought for. Freedoms cannot be won without a path to be forged.

### And we REJECT...

- **the growing necessity to rely on uncontrollable, unaccountable and unsecurable platform services.** There must always be an "off" switch. There must always be an "opt out." There must always be an option to secure your data from anyone. This can only be granted via absolute encryption or the decentralization of these platform services.
- **the default culture of complete and uncontrolled exposure that exists on the Internet.** Whether enforced by government will or corporate greed, the notion of having to "opt in" to privacy must be vigorously opposed. In order to fight government monitoring and capitalist profiteering on our sensitive data, the Internet must be more decentralized and the monopoly of data control must be broken.
- **governmental and corporate control over communication.** As stated before, governments and corporations cannot be trusted to act as humanity's intermediaries. Any method by which a government can extrajudicially monitor communications must be resisted. Any method by which a corporation can enact a "paywall" to knowledge and exploit class divisions in society must be resisted.
- **centralized communication platforms of control and oversight.** Any platform that allows our communications to be easily intercepted is, at the end of the day, an enemy to truly free expression.
- **software and tools that are "closed source," not hackable or not open for public inspection.** Whether its intended to aid capitalist competition or to serve as a weapon against others, closed source software is not acceptable on an open Internet.
- **the taking advantage of a user's technical ignorance for personal gain.** The lack of education regarding secure communications and encryption for the common user must be rectified if we are to see any substantial change. Proliferation of easy tools to ensure secure/private communication must be given the highest priority.

## Therefore, we RESOLVE...

- **to force governments and corporations around the world to hear our voice.** We refuse to play by your rules. We refuse to live in your walled gardens. We refuse to give our personal lives over to you for your profit. We will create the Internet that we want, and will communicate how we like.
- **to work with one another to build the next generation of the Internet.** The technical obstacles to decentralization remain high. Through the development, education and testing of new software and technologies, we can bring ourselves over this roadblock and help create a better world.
- **to resist, in whatever manner we are capable, the centralization of the Internet, and the bulk, indiscriminate monitoring it is accompanied by.** Whether this be through the general encryption of our data whenever possible, the forced removal of our accounts from the large platform services, or a mixture of the two, we will do our best to stand in the way.

## ***The CitizenWeb Guides – Getting Started with Linux***

### **2.1. Choosing a Distribution**

---

#### **2.1.1 - What do I need?**

Choosing a Linux distribution may seem like a daunting task. In fact, there are hundreds of distributions out there; dozens of them worthy contenders for most computers. However the ability to choose between them has improved remarkably in recent years.

Ask any Linux user "What distro should I use?" and the answer will most likely be "go with what you need." Every distribution has their strong points and their weak points. To begin, make a list (mental or otherwise) of what you seek to accomplish with your computer:

- **What will I work on with this computer?** If this is primarily to be an internet and office work machine, most any distribution can do that with relatively little configuration. However more advanced programs will require distributions with better codebases and well-maintained repositories.
- **What is my skill level?** Those who are just starting Linux for the first time will most likely want to choose a more "simple" distribution. And there are plenty of them: built for ease of use, compatibility and clean user environments right off the bat. For those who are looking for a challenge, and would like to customize their system for power and speed, an "advanced" distro might be more to their liking.
- **How much do I want to configure my visual interface?** Linux has no shortage of decent graphical environments, known as "Desktop Environments" and "Window Managers." The distribution you choose will largely depend on which graphical environment suits you. Many of the newer, more simple distributions like Ubuntu and Linux Mint, have specific editions depending on the environment you want to use. In any Linux distribution there is the freedom to set your own DE/WM; however if one prefers XFCE for example, they are better off downloading Xubuntu over the standard Ubuntu distribution.

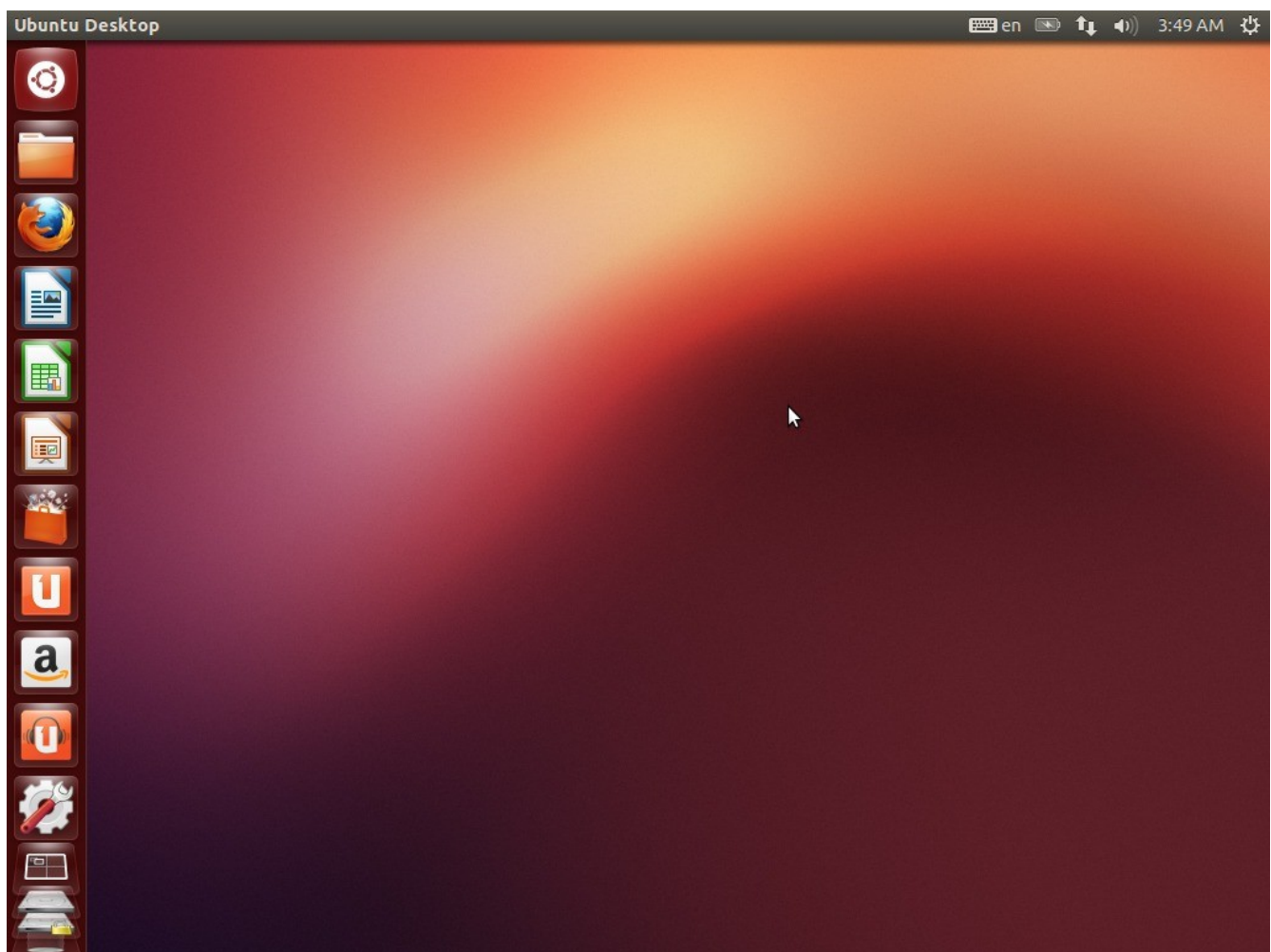
### 2.1.2 - The Distros



This is by no means an exhaustive list of Linux distros; only a list highlighting the most popular choices. For a more detailed list and comparison, visit [Distrowatch](#).

The distros here are listed by their general ease-of-use and ease of install; Ubuntu being the easiest and Arch the most difficult. The inverse is true for the amount of say you have in packages installed by default: Arch is most customizable in this regard, while Ubuntu is the most restricted.

## Ubuntu



- **Website:** <http://ubuntu.org>
- **Package management system:** aptitude (apt-get)
- **DE Versions:** GNOME/Unity (default); other versions come via offshoots
- **Derivatives/Editions:** [Xubuntu \(XFCE\)](#), [Kubuntu \(KDE\)](#), [Lubuntu \(LXDE\)](#), [Crunchbang \(Openbox\)](#)
- **Pros (from Distrowatch):** Fixed release cycle and support period; novice-friendly; wealth of documentation, both official and user-contributed
- **Cons (from Distrowatch):** Lacks compatibility with Debian; frequent major changes tend to drive some users away

Ubuntu (and its derivatives) is the most popular choice of distribution for Linux users. It is very easy to use, giving users the option of using the system without meddling with the command line at any point. This grants the user with an experience similar to Windows and Mac OS X. In these respects, Ubuntu is the "easiest" distribution to get into and to learn, and is a great choice for beginners. With an emerging hold in the business and server market, Ubuntu is seen as being a stable and consistent option as far as distributions are concerned, with a company (Canonical Ltd) in charge of its development and maintenance. While recent releases have not quite lived up to its own high standards it has achieved in the past, Ubuntu remains a solid choice and a logical conclusion for Linux beginners.

## Linux Mint



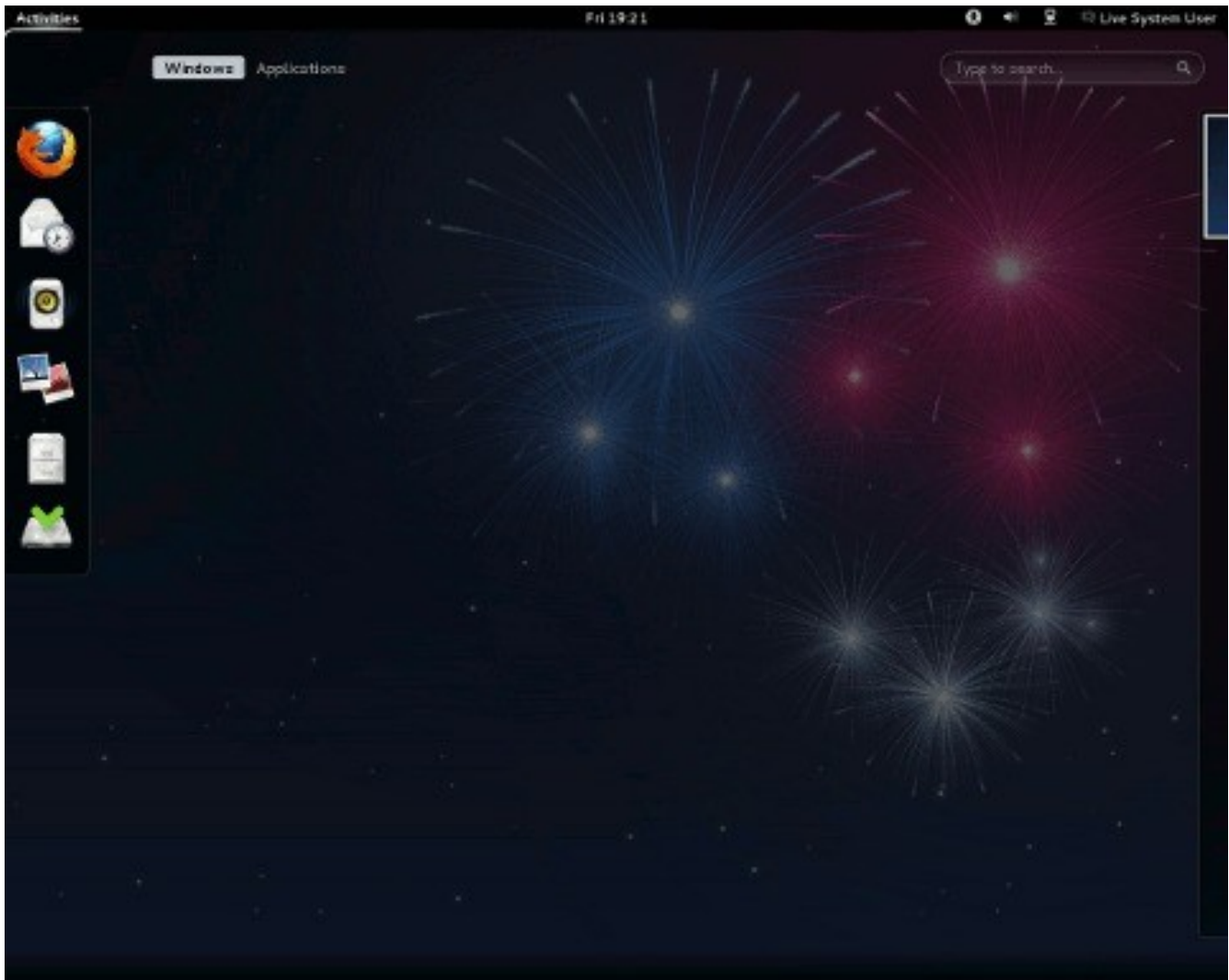
- **Website:** <http://linuxmint.com>
- **Package management system:** aptitude (apt-get)
- **DE Versions:** Cinnamon (default), MATE, KDE, XFCE
- **Pros (from Distrowatch):** Superb collection of "minty" tools developed in-house, hundreds of user-friendly enhancements, inclusion of multimedia codecs, open to users' suggestions
- **Cons (from Distrowatch):** The alternative "community" editions don't always include the latest features, the project does not issue security advisories

Linux Mint originally began as a derivative of Ubuntu. It is maintained by a community that wanted to take some features of Ubuntu in new directions. The most notable difference between Mint and Ubuntu is its readily-enabled freedom to choose one's own graphical



(desktop) environment. Other than that, both Ubuntu and Mint are based off of Debian, making them closely related systems in terms of maintenance and preferred software suites. Mint also includes its own suites of software to manage specific functions, which adds to this distribution's ease-of-use.

## Fedora



- **Website:** <http://fedoraproject.org>
- **Package management system:** yum
- **DE Versions:** GNOME (default), KDE, LXDE, XFCE
- **Pros (from Distrowatch):** Highly innovative; outstanding security features; large number of supported packages; strict adherence to the free software philosophy; availability of live CDs featuring many popular desktop environments

- **Cons (from Distrowatch):** Fedora's priorities tend to lean towards enterprise features, rather than desktop usability; some bleeding edge features, such as early switch to KDE 4 and GNOME 3, occasionally alienate some desktop users

Fedora is the community-run stepchild of one of the oldest and most well-known Linux distributions, Red Hat Linux. Now that Red Hat is only available for enterprise applications, Fedora is the distribution that is being offered to general end users. Fedora is different from both Ubuntu and Linux Mint in that it is not based off of Debian; therefore it uses a different package management system as well as its own suite of applications and services. Fedora is considered to be a stable and mature distribution, perhaps not with the same ease-of-use that Ubuntu provides, but is not far behind. It is a decent choice for intermediate computer users, as well as beginners to Linux looking for more of a challenge.

## Arch Linux



- **Website:** <http://archlinux.org>
- **Package management system:** pacman
- **DE Versions:** Any (installed custom)
- **Pros (from Distrowatch):** Excellent software management infrastructure; unparalleled customisation and tweaking options; superb online documentation
- **Cons (from Distrowatch):** Occasional instability and risk of breakdown, infrequent install media releases

Arch Linux prides itself on its core philosophy: "Keep It Simple, Stupid!" In line with this idea, Arch tries to keep its distribution as clean and free of unnecessary clutter as possible. While systems like Ubuntu include resource-heavy front-ends like the Unity window manager and many application suites installed by default, Arch prefers to let the user choose what they want their system to be by default. This way allows for maximum customization and minimum time lost working with conflicting or unused and bloated software tools. Arch also differs from most other distributions in that it prefers a rolling-release style; where other distributions each have versions and releases of their software, Arch stays on the cutting edge by providing all updates through `pacman` once they are available.

These characteristics admittedly makes Arch one of the hardest Linux distributions to install and maintain, as everything must be selected by the user, installed and maintained without the kinds of blueprints that other distributions might offer. However the Arch community is very friendly, close-knit and features an amazing Wiki full of documentation. Arch is a great choice for power-users or those looking for a serious challenge with maximum reward and customization opportunity.

## 2.2. Installing Ubuntu

---

### 2.2.1 - Downloading and Burning Ubuntu

Installing Ubuntu is a breeze, made easy by its convenient graphical installer that rivals the ease-of-use of either Microsoft or Apple's operating systems.

First, you'll need to download and burn the ISO file. Go to [ubuntu.com](http://ubuntu.com) and click Download. Choose "Ubuntu Desktop." Choose the "For the latest features" option, then pick the correct architecture in the "Choose your flavour" box. Then click the Get button. You may be presented with a screen to solicit donations: make one if you'd like, OR scroll to the bottom and choose "No thanks." The file will download automatically.

Once the download is complete, you'll need to load a blank disc into your computer. The next steps depend on the operating system you are using.

- **Windows 7:**
  - Double-click the ISO file you downloaded to open the "Windows Disc Image Burner."
  - Click "Burn."
- **Windows XP (or older):**
  - Download imgBurn from <http://www.imgburn.com/>.
  - Open imgBurn and choose "Write image file to disc"
  - Select the ISO you downloaded and click "Burn."
- **Mac OS X:**
  - Open the "Disc Utility" application in Applications > Utilities.
  - Drag the ISO file you downloaded to the left-hand sidebar. Select this file and click "Burn."

### 2.2.2 - Prepare Your Computer and Files

Once you've burned Ubuntu to disc, you will need to prepare your computer for your Ubuntu install. This will depend on your desired setup:

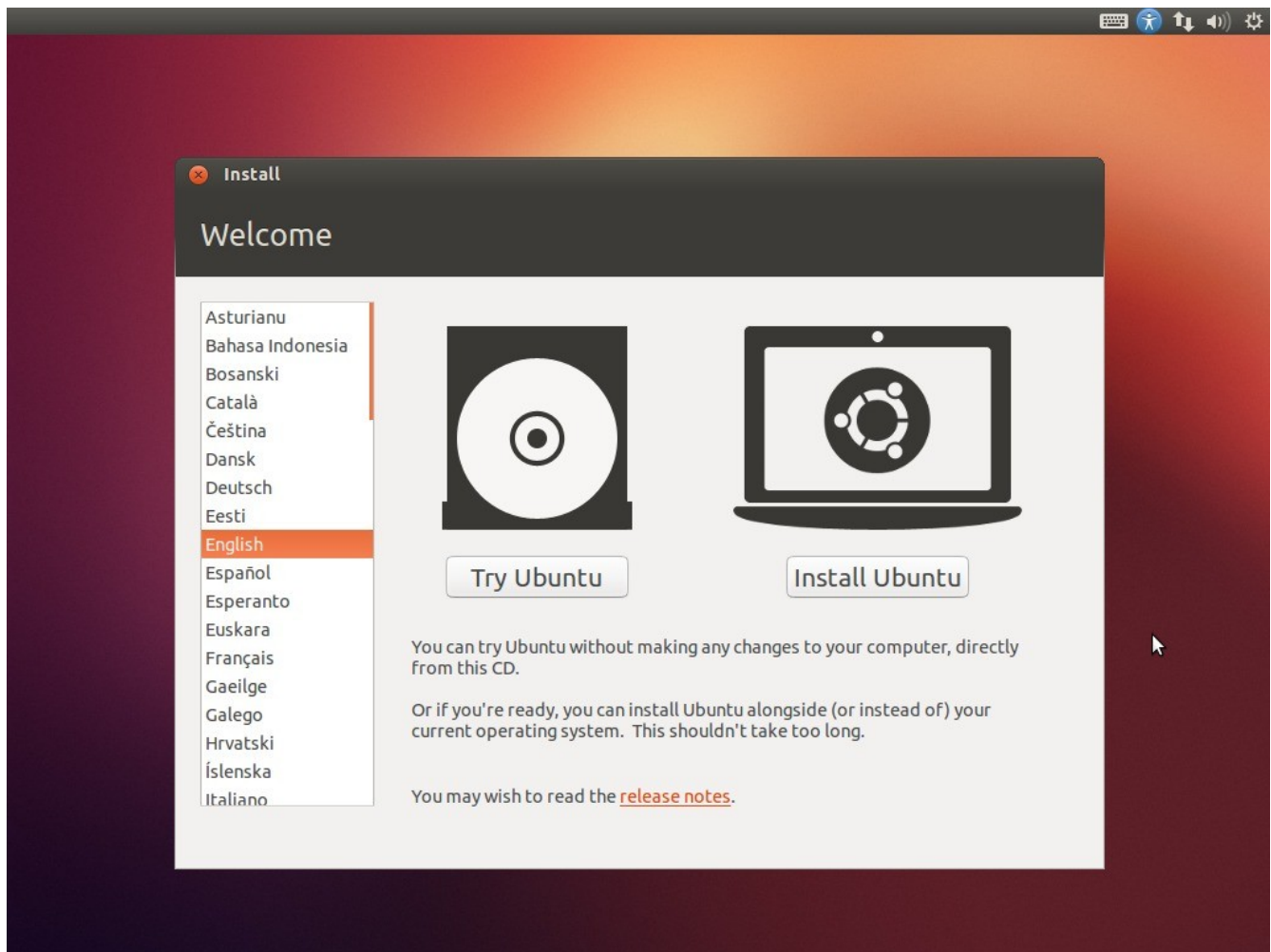
- Most users will want to ONLY use Ubuntu as their sole operating system. For this, no extra prep is required.
- For those who wish to (or need to) use Windows as well, AND have a computer new enough, they can opt for a full install of Ubuntu and then to use a Virtual Machine to run the programs they need for Windows. No extra prep is required for this step either. *(Keep in mind that you must have a valid Windows install disc to choose this option.)*
- For those who wish to (or need to) use Windows as well, but don't have a fairly-new computer with a multi-core processor, they can opt for a multi-partition setup. This consists of a sole computer with two operating systems installed on it, and the OS to use can be chosen at boot. So if you have both Ubuntu and Windows installed, and you want to switch to the other operating system for awhile, you can simply reboot your computer and switch at the bootscreen. If you wish to use this option, keep an eye out for the "dual-boot setup" option in the Installation section. *(Keep in mind that you must have a valid Windows install disc to choose this option.)*

No matter what you have chosen above, you will need to erase your entire hard drive (unless your hard drive presently has enough unpartitioned free space on it, which is doubtful). Before you do this, make sure to back up all of your files to external USB drives or disks. Keep them safe until you can offload your data onto your computer again.

### 2.2.3 - Installing Ubuntu

Load your Ubuntu install disc into your computer and reboot. The computer should boot from disc automatically. If it doesn't, visit your computer manufacturer's website and look through the support section for how to boot from disc.

On boot, Ubuntu will load an interface from CD, then present you with this lovely screen:



If you'd like to try the interface out a bit before you begin, feel free to click "Try Ubuntu." You will be able to go to the installer via a link on the desktop. When you are ready to install, click "Install Ubuntu."

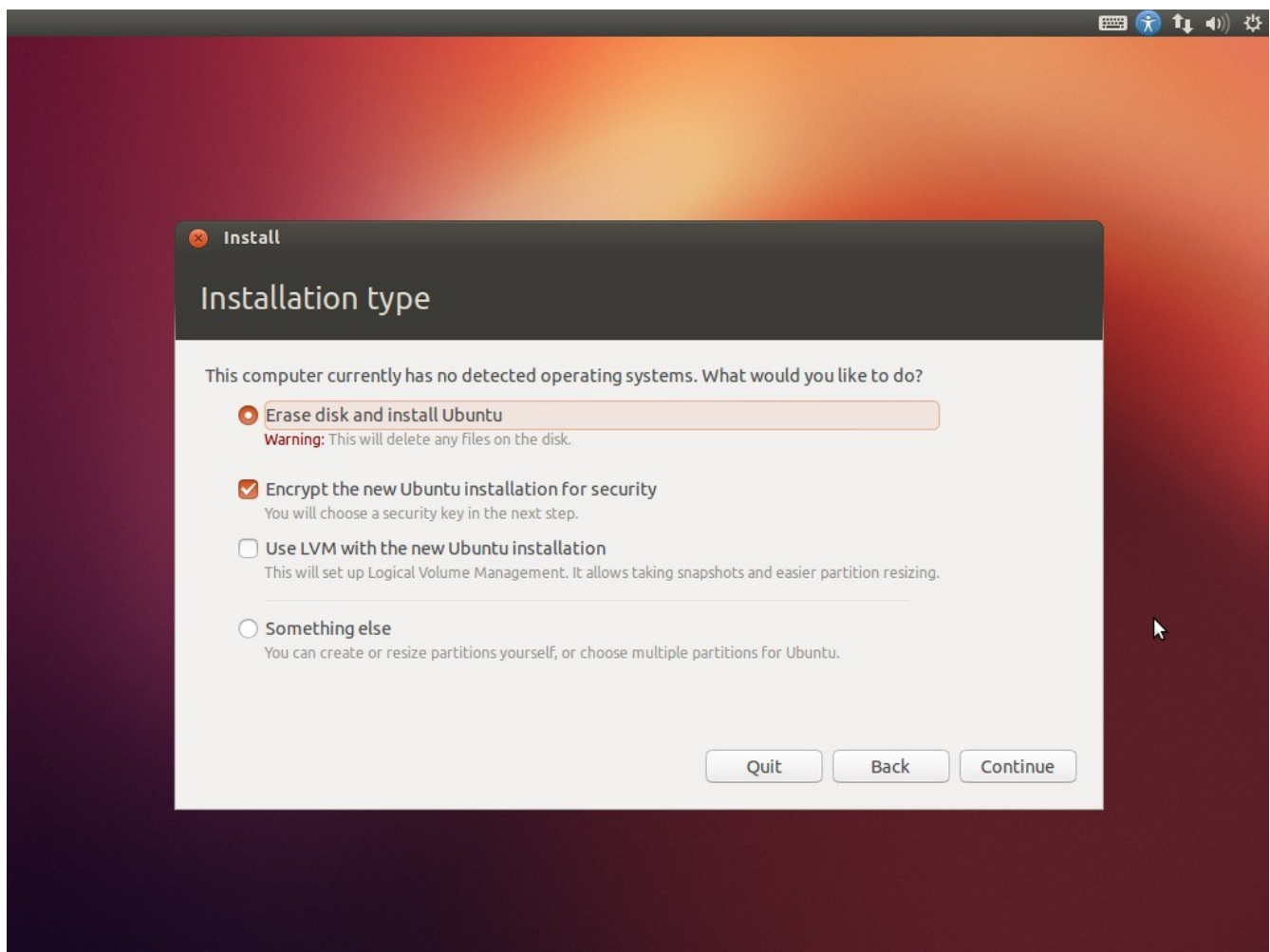


Don't be afraid if Ubuntu seems really sluggish here before you install it - after all, it's running from your CD drive which is many times slower than your actual hard drive will run!

Check "Download updates while installing" then click Continue.

If you wish to use Ubuntu as your sole operating system, choose "Erase disk and install Ubuntu." If you wish to use a dual-boot setup as explained above, click "Something Else," which will take you to a partition management screen. (At this point you should see the Partitioning section below).

If you decided to install Ubuntu with full-disk encryption, check "Encrypt the new Ubuntu installation for security" and click Continue. The next window will provide you with an opportunity to choose your security key. It's recommended that you choose to "overwrite empty disk space," especially if this is not a new computer.



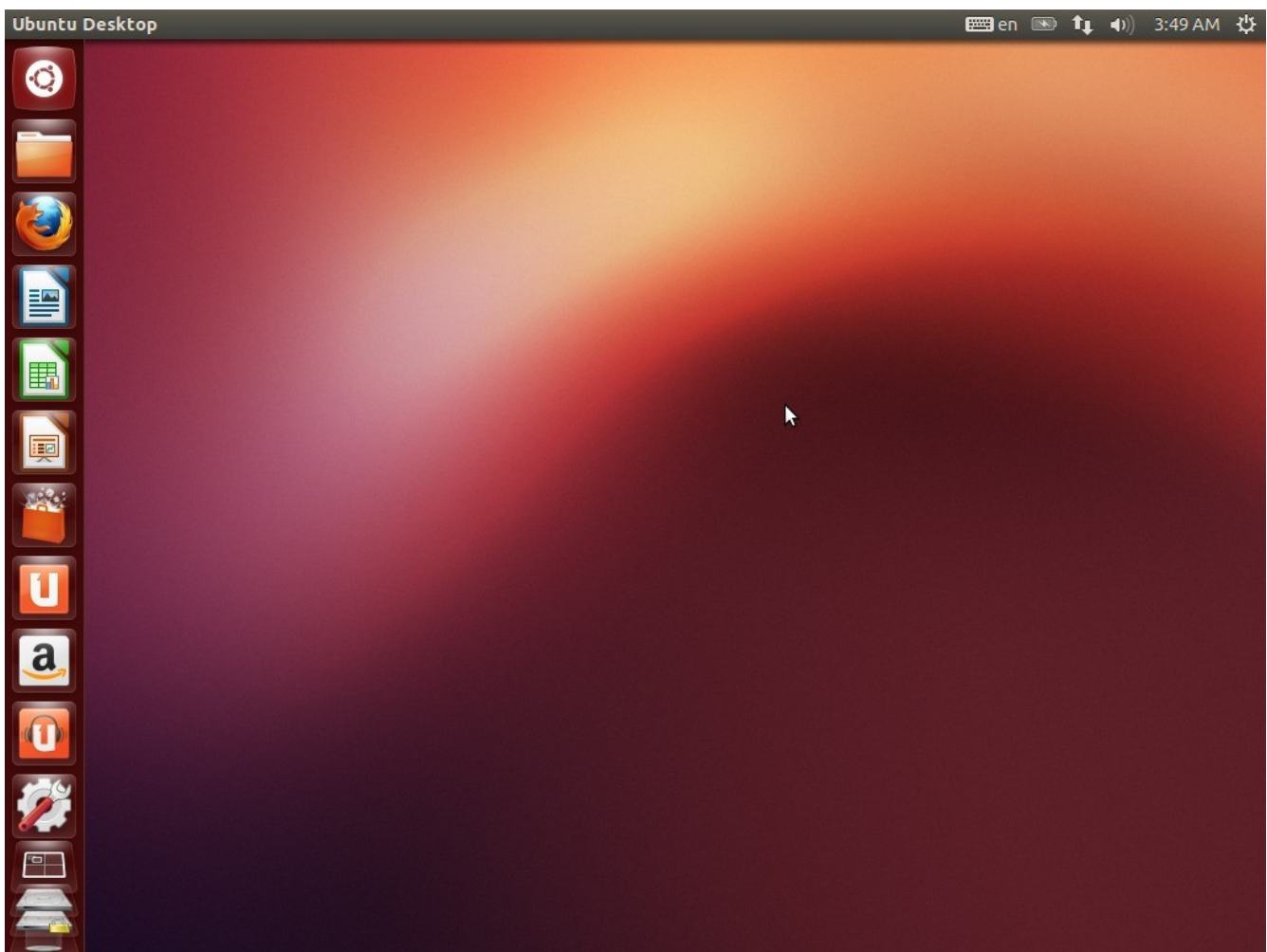
While Ubuntu installs, the next screens will give you the option to choose a variety of options, including your timezone, preferred keyboard layout, and credentials. Once that's done, sit back and enjoy the wait.



Once Ubuntu reboots itself, you will be put at your login prompt, then the desktop. You made it!

### 2.2.4 - Getting Used to Ubuntu

Ubuntu is one of the easiest Linux distributions to use. It's perfect for users looking to set up their computer with minimal tweaking and configuration.



Ubuntu's primary interface is called "Unity." You'll see that the desktop has a bar on the upper edge of the screen, which is where your notifications and your menu bar for applications will pop up (Mac OS X-style). Along the left-hand side of your screen you will see the Dock. This has icons of frequently used applications that can easily be launched from



here (again, like Mac OS X's Dock). You can add or remove programs to the dock by simply clicking and dragging them to or from the dock.

Unity's (arguably) best feature is the Search pane (similar to Mac OS X's 'Spotlight' - do you see a pattern here? :) ). This is the top magnifying glass-shaped icon on the dock. Click here and you can browse your applications and your files depending on their type. It's fairly intuitive and shouldn't take too long to figure out. There is also a search box at the top where you can enter part of a filename or application name, and it will bring that object up for you to load.

In the Search box, type "term" and click the Terminal icon that comes up. This is your standard Linux command line terminal. We will be using this often for configuring the client and setting up software. The good thing about Ubuntu is that there are graphical alternatives for establishing almost any setting - however it's better to work from the command line when one is learning, to better gain a grasp of what exactly is going on beneath the applications you are configuring. In this guide, graphical alternatives will be mentioned when they are available, but we will always be working from this terminal.

On the dock, you will notice a picture of a gear and wrench. This icon opens the System Preferences screen, which will allow you to customize your system to your heart's content. If, for example, your mouse seems a bit faster here than it did in Windows? Go to the Mouse section and you will be able to adjust it to meet your needs. Feel free to play around with this before we get into the nitty gritty of setting up your system.

### **2.2.5 - Dual Boot Partitioning (Optional)**

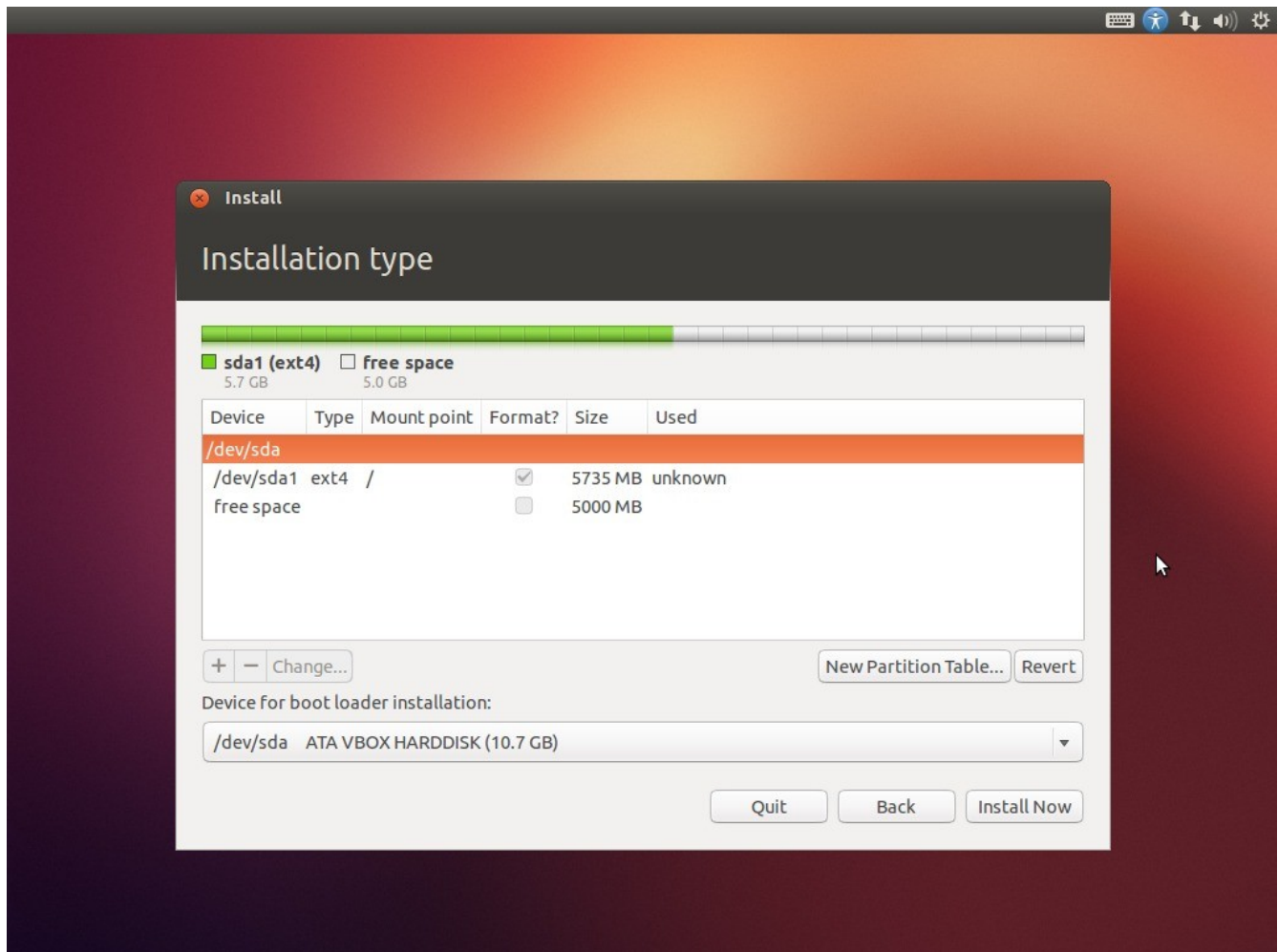
If you need to keep a Windows installation on your hard disk (and are unable to use a Virtual Machine), you can choose to set up a custom partition table during the Ubuntu installer.



Note that you cannot use a custom partition table AND use full-disk encryption in the Ubuntu installer at present.

First, delete all existing partitions (anything with a number after the "/dev/sd?" bit) by selecting them and clicking the "-" button. Then, to create a new partition, click the "+" button. You will be able to define the partition's size in megabytes (1,024 MB = 1 GB), as well

as select its filesystem type and mount point. For the main partition, set it to the size you wish and set the mount point at "/". Linux partitions should be set to use the ext4 file system.



For the Windows partition, just leave some "free space" that matches the size of the Windows partition you wish to make. When you load your Windows disc installer, you will create a partition in this free space and choose to install Windows here.

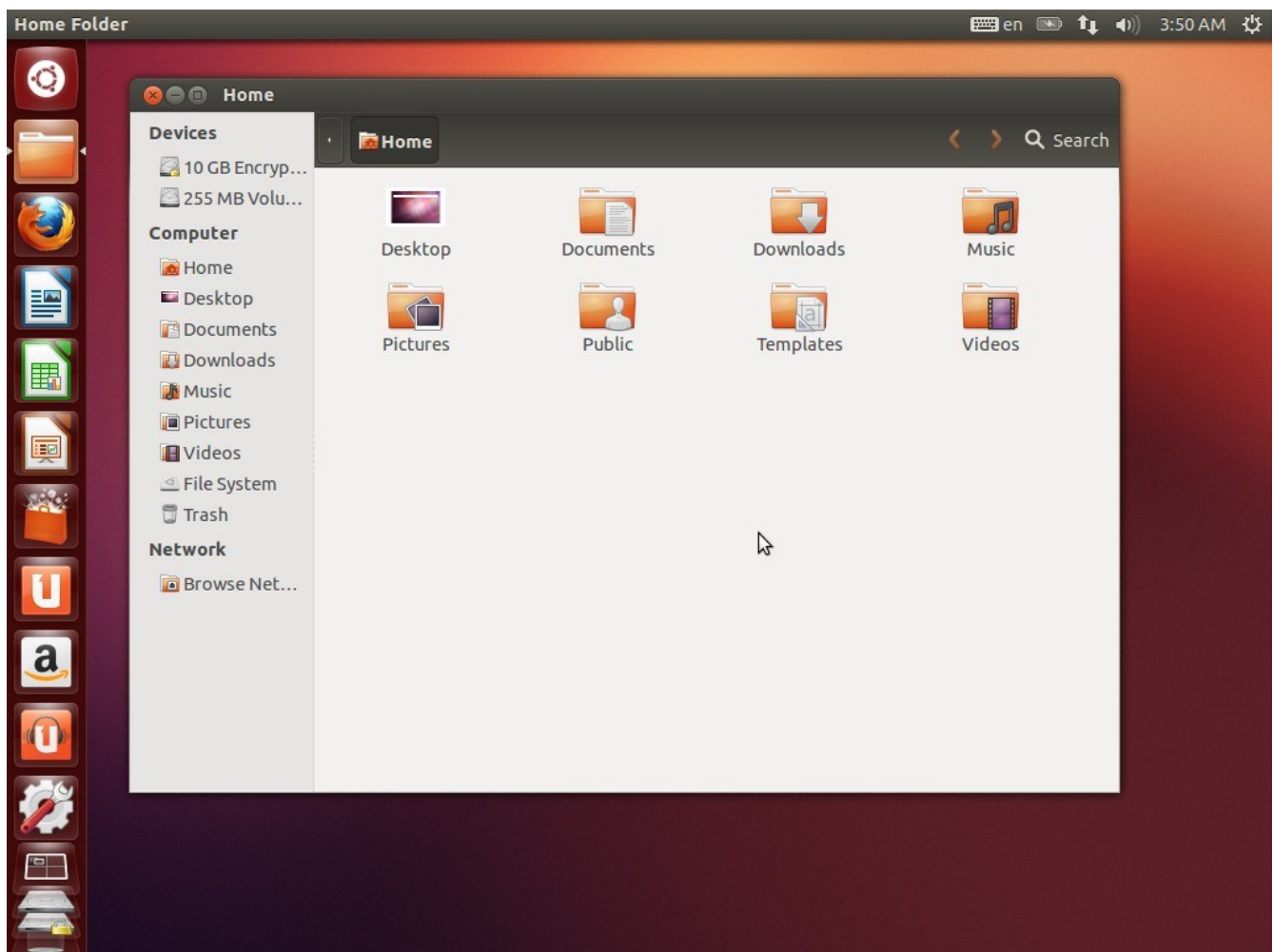


Keep in mind that Windows requires a lot more space to operate than Linux does. For Windows you should look to set aside (at a bare minimum) 50GB of space for the operating system and some application suites.

## 2.3. Getting Used to Ubuntu

### 2.3.1 – The Ubuntu Experience

As explained before, Ubuntu's main interface is called `_Unity_`. The menu bar is along the top of the screen, where you will be able to see the standard File, Edit, Window, and other menu buttons. This is much like the functionality of Mac OS X. Towards the right side of the menu bar, you find options based on the applications you are running, as well as the standard tray icons (Network, Volume, Settings) and the system time.



Along the left side of the screen, you see Ubuntu's version of the Windows Start bar or the Mac OS X Dock. This dock shows you your frequently used applications. You can pull applications to this Dock for quick reference, or remove them simply by pulling them off the Dock.

The first button on the Dock (with the Ubuntu logo) brings up the Search pane. This is the second most convenient way to launch applications in Ubuntu. The Search pane is your center for finding programs and files on your hard drive. You can type the first few letters of the application you are looking for, and it will come up at the top of your search. You can also type the name or other details about a document/file you are looking for on your hard drive, and the Search pane will look for it for you. At the bottom of the pane, you can see some buttons to filter your searches. You can choose to search only for applications, documents, music, photos, or video. The Search pane also allows you to search for products for sale on Amazon.com (though this can be turned off in System Preferences > Privacy).

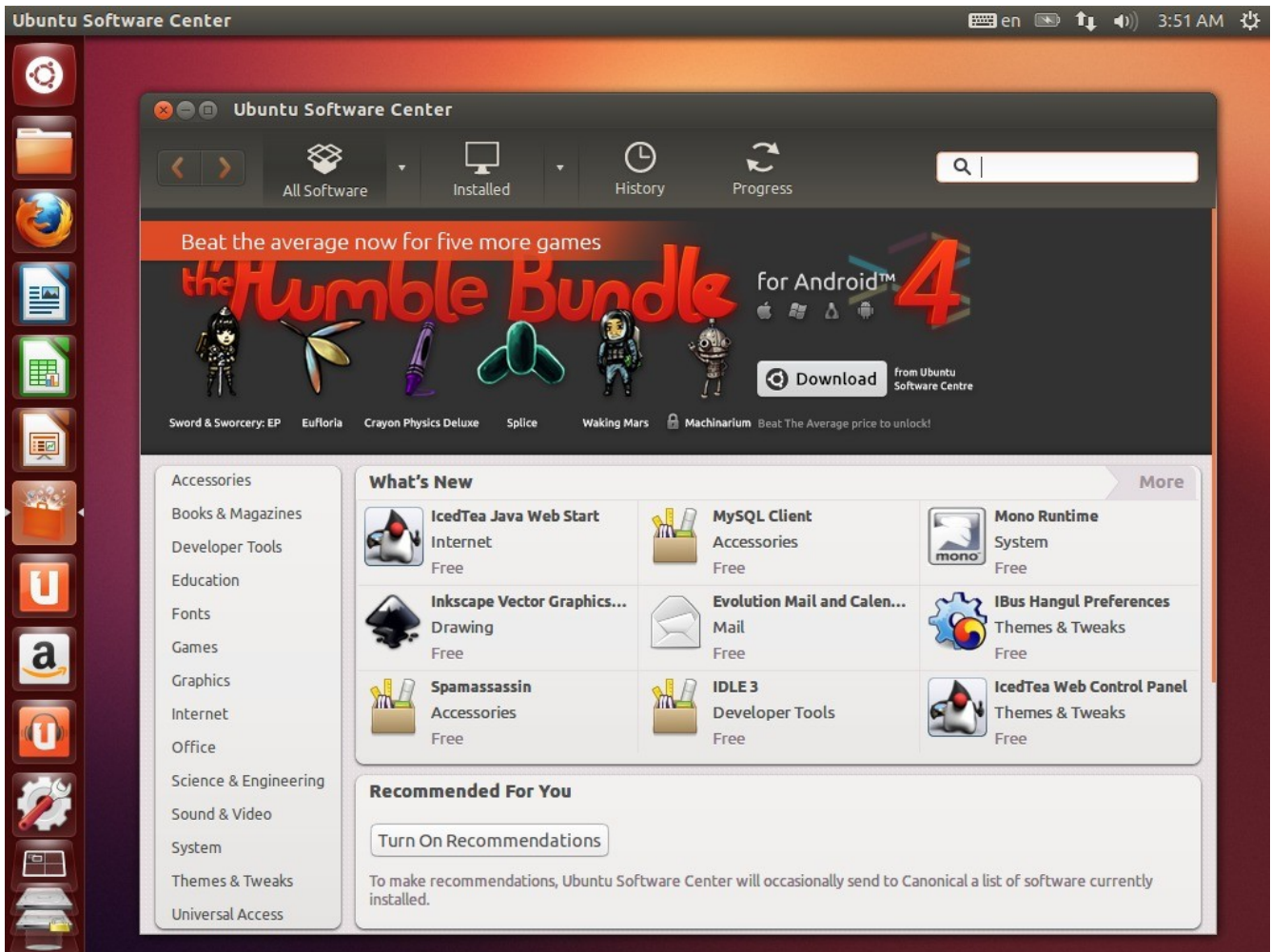
The second button in form of a file folder is your File Explorer. This is analogous to the Windows Explorer (or clicking "My Documents") in Windows, or the Finder in Mac OS X. As you can see by the photo above, the interface is very similar to both of these other desktop environments.

### **2.3.2 – Applications and Features**

The next app in the list by default is Firefox. This runs the popular Internet browser.

Next, we see three document icons. These run LibreOffice, an office suite similar to Microsoft Office, but open source and centered around open-source file formats. LibreOffice is very intuitive and easy to use. The icons represent Writer for word processing, Calc for spreadsheets, and Impress for creating presentations.

Ubuntu has a center for finding new programs and utilities you might find useful, called the Ubuntu Software Centre. The Software Centre is identified by the picture of the shopping bag in the Dock. Here you can find apps in a wide variety of categories, free or paid. Most of them are actually free. You can manage software you've installed, uninstall old packages, or manage system updates from the Software Centre.



Next is the Ubuntu One logo. Ubuntu One is a cloud solution provided by Canonical (Ubuntu's parent company). It is similar to Google Drive. You can sign up for a free account to store your music, photos and documents online, then access them from anywhere in the world on a variety of different platforms. There are also paid options that unlock some additional functionality.

Last on the Dock list for now is the System Preferences pane. This is indicated by the picture of the gear-and-wrench in the Dock. Here you can customize some of your system's most important features, like language, date/time, privacy settings, network preferences, and more.

Other applications you will find of interest, but that may not be in the dock:

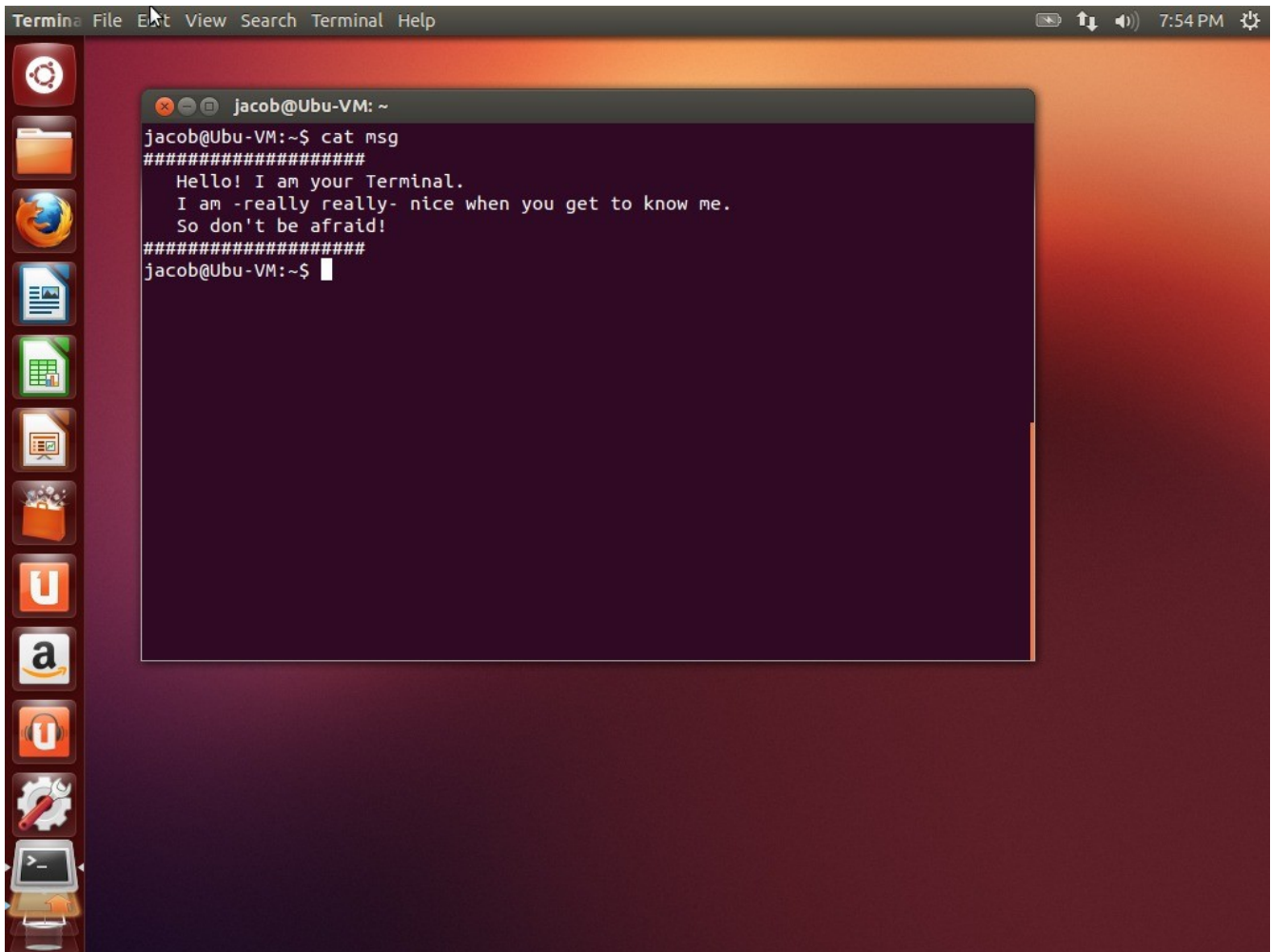
- **Rhythmbox** - This is Ubuntu's default music player. Similar to iTunes, it plays your music and manages your library with a clean and intuitive interface.
- **Thunderbird** - This is Mozilla's mail client, much like Mac OS X Mail, or Microsoft Outlook.
- **Text Editor** - Fancy a quick note? Use this application, analogous to Notepad on Windows or TextEdit on Mac OS X.
- **Calculator** - Self-explanatory!
- **Shotwell Photo Manager** - A free photo library manager, very similar to Mac OS X's iPhoto.

Head further into the Dock and the Ubuntu Software Centre, and see what neat applications you can find! Or go to chapter 2.5 in the Guide to get a list of more applications that may be helpful.

### ***2.3.4 – A Brief Introduction to the Terminal***

The bane of every new Linux user is the Terminal. However it is mostly much ado about nothing. With Ubuntu, you can use Linux on a day-to-day basis without even needing to touch the terminal. And its function is surprisingly simple when it comes down to accomplishing basic tasks.



A screenshot of an Ubuntu desktop environment. The desktop has a dark red and orange gradient background. On the left side, there is a vertical dock with several application icons: Dash, Home Folder, Firefox, LibreOffice Writer, LibreOffice Calc, LibreOffice Impress, a folder icon, the Ubuntu logo, Amazon, another Ubuntu logo, a settings icon, and a terminal icon. The top of the screen shows a menu bar with 'Terminal', 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The system status area on the right shows network, volume, and battery icons, along with the time '7:54 PM'. A terminal window is open in the center, titled 'jacob@Ubu-VM: ~'. The terminal shows the command 'cat msg' being executed, which displays a message: 'Hello! I am your Terminal. I am -really really- nice when you get to know me. So don't be afraid!'. The prompt 'jacob@Ubu-VM:~\$' is visible at the bottom of the terminal window.

```
Terminal File Edit View Search Terminal Help
jacob@Ubu-VM: ~
jacob@Ubu-VM:~$ cat msg
#####
Hello! I am your Terminal.
I am -really really- nice when you get to know me.
So don't be afraid!
#####
jacob@Ubu-VM:~$
```

When you launch the Terminal, you begin in your Home directory. You can tell this by the tilde (~) in the command prompt. Your location in the hard drive will always be given in this space.

To list the contents of the directory you are currently in, type ``ls`` and press Enter (#1). You just ran your first command via the Terminal!

To navigate to a different folder, run ``cd`` and follow that with the folder name. As you can see below, I ran ``cd Documents``, and it put me in my Documents folder (#2). Simple enough. When you want to go back to the folder (like the "Up" button in Windows Explorer), run ``cd ..`` and you will be taken back (#3). You can run these commands via absolute paths, i.e. folders that are not in the folder you are currently in (#4, #5).

```

Terminal
jacob@Ubu-VM: /usr/share/backgrounds

jacob@Ubu-VM:~$ ls 1
Desktop  Downloads  Pictures  Templates  Videos
Documents Music      Public    Ubuntu One

jacob@Ubu-VM:~$ cd Documents 2
jacob@Ubu-VM:~/Documents$ ls
File1 File2 File3 Folder1 Folder2

jacob@Ubu-VM:~/Documents$ cd .. 3
jacob@Ubu-VM:~$ ls
Desktop  Downloads  Pictures  Templates  Videos
Documents Music      Public    Ubuntu One

jacob@Ubu-VM:~$ cd /usr/share/backgrounds 4
jacob@Ubu-VM:/usr/share/backgrounds$ ls
A_Little_Quetzal_by_vgerasimov.jpg      Green_Plant_by_Simon_Schlegl.jpg
Below_Clouds_by_kobinho.jpg              H_by_Manuel_Sagredo.jpg
Cairn_by_Sylvain_Naudin.jpg              Pantano_de_Orellana_by_mgarciaiz.jpg
contest                                  Roof_Tiles_by_Finn_Sturdy.jpg
Early_Morning_by_Robert_Katzki.jpg        Vanishing_by_James_Wilson.jpg
Frozen_by_fernando_garcila_redondo.jpg    warty-final-ubuntu.png
Gran_Canaria_by_ALF.jpg

jacob@Ubu-VM:/usr/share/backgrounds$ ls /home/jacob/Documents 5
File1 File2 File3 Folder1 Folder2
jacob@Ubu-VM:/usr/share/backgrounds$

```

This is the basics for navigating through folders in the Terminal. For file manipulation, you can follow the same process for putting command + file location together. Copying files is achieved with ``cp``, followed by the file you want to copy, then its location. So: ``cp sourcefile.txt /home/user/Documents/`` will copy the "sourcefile.txt" in the current folder to your Documents folder. In the same way, you can use "mv" to move, or "rm" to remove files. You can also use "mkdir" to make new folders.

Beyond simple file management, using the Terminal can be boiled down to one simple fact: **terminal commands are applications plus options**. Every application that you run on Linux has a corresponding Terminal command that can be used to run it. Furthermore, these commands can use option "flags" to adjust its manner of operation.



To explain this, let's take a look at a basic command called `tar`. TAR is used to create archives of files or folders, much like the ZIP file format on Windows. To create a standard zipped-up TAR archive of a file, we run the following command:

```
tar -cvzf archivename.tar.gz filename.ext
```

This creates an archive named "archivename.tar.gz" that contains the file "filename.ext". But what about those letters following the `tar` command? Those are the flags. In Linux, flags are denoted with the "-" that comes before them, and usually come right after the initial command in the string. If you want to use more than one flag, you can stack them, like I did above, with just one "-".

Let me explain what each of those flags does for this specific `tar` command:

- **c** means to (c)reate the archive. You can also use TAR to extract from existing archives, so that is why you must specify that you wish to (c)reate one.
- **v** means to output (v)erbosely. In plain English, this tells TAR that we want to see everything it is doing, as it does it. So it will then print out a list of each file it is compressing as it does so. Or, if it gets an error, it will give us a full readout of where the error occurred.
- **z** means to g(z)ip the TAR archive. This adds a level of compression to our archive, so their contents will be smaller than they would be outside of the archive. Just like a ZIP file on Windows.
- **f** means that we will specify the (f)ilename of the archive we will make. Otherwise, TAR will automatically generate a name for our archive.

Each command you will want to use on the command line has a corresponding "manpage." (It might sound like a sexist name, but it just means "manual"!) Here you can get detailed information on how to use the command, as well as a list of flags and commonly-used options for it. Simply run `man` plus the command you want to learn about. `man tar`, for example, will show you the manual and flags list for the `tar` command.

There, the Terminal is easy, just like I told you! It might not seem very convenient at first, but the more you get to use it, the quicker frequent tasks can pass by, leading to great increases in your productivity. For example, to create a TAR archive of a file (or folder), by the standard way you would need to launch the Archive Manager application, mouse over to

"New Archive," click it, type in a name, type in a place for the archive to be, mouse over the checkboxes for options, click and drag your folders, etc etc etc. But with the Terminal, after learning how the command works the first time, you can simply run a quick command from memory to do exactly what you want. You can even create scripts (called "bash scripts") to automate tasks using the Terminal's language. We will cover this in a future guide. But for now, pat yourself on the back, because you've conquered your fear of the Terminal!

## 2.4. Securing Web, Email and Chat Applications

---

### 2.4.1 - Secure Your Web Browsing

#### Encrypt Your Connections with SSL/TLS

The first step to take in assuring your web browser's security is to make sure every connection possible is made over SSL. SSL should be familiar to you by now -- every time you log into your bank account, for example, you should see a little "https" in your address bar with a little green check-mark or a lock symbol. This means that your personal connection data is being encrypted between you and the server you are communicating with. Your username, your password and other form data on the bank's website cannot be "snooped" on by anyone else on your network.

Most sites that require logons will have SSL capability. The problem is that SSL is often not equipped by default on sites that don't handle financial information. This means that sites like Facebook might still be handling your connections over regular unencrypted HTTP by default.

To change that, there are browser plugins that you can use to enforce SSL by default for any site that has it enabled. The Electronic Frontier Foundation has developed a tool named "HTTPS Everywhere" which enables HTTPS by default on many popular sites and services that have HTTPS available. It can be paired with another plugin called HTTPS Finder or KB SSL Enforcer, which searches other sites that appear to have HTTPS installed, and passes them to HTTPS Everywhere.

HTTPS Everywhere can be downloaded for Firefox and Chrome (Chromium) by [visiting the website](#). In Firefox, click the Install link, then click Allow, then click Install Now. You will need to restart Firefox before the plugin will be enabled. For Chrome, click the Install link, click "Add to Chrome," then click "Add."

To install HTTPS Finder in Firefox, open your browser than click Tools > Add-ons. Click "Get Add-ons," and search for HTTPS Finder.

To install KB SSL Enforcer in Chromium: go to the [Chrome Web Store](#) and search for "KB SSL Enforcer" under "Extensions."

And in the future - for EVERY site that requires a login, make sure that the address shows as HTTPS on the login page! If not, go into your settings and there will often be an option to use HTTPS by default tucked away somewhere.

## Block Monitoring Scripts

Once your connection data is secured, there is now the matter of tracking scripts. Everywhere on the web, on nearly every commonly-used website nowadays, there are "trackers." Trackers come in many forms but the most frequent way is via tracking cookies that are transparently downloaded to your computer, or active scripts that run on pages that report home with specific data. The idea of web tracking is a very broad concept but I will give two of the most common examples below.

1. Google Analytics -- This is a piece of software run by Google that gives webmasters a huge amount of data on each visitor. It can pinpoint everything from their approximate geographic location, to details about their computer and its operating environment, to how long they spent on the site, what pages they looked at, and other details. It can even tell what site you came from to get to a certain location, and what site you visit when you leave. Some of this information can be discerned simply by looking at one's server logs, but Analytics renders this much easier. Not everyone has a problem with this data being in the hands of webmasters, but the fact that it is also kept by Google can be more than worrying.
2. Facebook -- Facebook's "Beacon" scripts are everywhere on the web. Anywhere you see a dynamically-loaded "Like" button, on a blog article or on a company's website, this information is sent to Facebook. What's worse is that if you are logged into Facebook (which most people are, even if it is not actively open in their browser), Facebook will be able to match your profile with your Web browsing habits.

If you'd rather not surf the web with strange corporations watching your every move, you certainly aren't alone. Thankfully there are browser plugins that can help! There is one in particular called Ghostery that is very effective at blocking trackers you don't want to see, while still giving you the power to enable the ones you may find useful from time to time. If you like the ability to click the "like" button from time to time, for example, but don't want Google Analytics to track you, you can manually allow the Facebook tracker in Ghostery's easy-to-search database.

To install Ghostery in Firefox or Chrome, go to the browser's Add-ons section and search for Ghostery. Once it is installed, it will ask you what sites to block. My advice is to choose "Select All" to block trackers by default. Then, later on, if you find one you need to use, you can go back into your Add-on settings and uncheck the box next to that tracker's name.

### Blocking Options

Ghostery groups 3pes into categories. Click on a category row to browse 3pes in that category. Mark checkboxes to block, click element names for more information.

Note: Blocking may interfere with webpages in unexpected ways. If you experience issues with videos, logins, comment forms, etc., pause blocking by clicking on the Ghostery icon, or whitelist the affected page.

3pes
Cookies
Site whitelist

1311 total 3pes (1235 blocked)

When you block a 3pes, that element is prevented from communicating with its third-party provider.

Show all | Search for name | [Reset search](#)

Expand all | Collapse all | Select all | Select none

▶	Advertising	632 elements (588 blocked)
▶	Analytics	247 elements (241 blocked)
▶	<input checked="" type="checkbox"/> Privacy	18 elements (18 blocked)
▶	Trackers	272 elements (257 blocked)
▶	Widgets	142 elements (131 blocked)

Save
Cancel

With Ghostery you can also pause all tracking easily. If you find a website doesn't quite work properly without its trackers, click the Ghostery button in your browser window, than click the "Pause" button. Then refresh the page and try the functionality again. Just don't forget to press "play" again when you are done!

## Encrypt Your Browsing with Tor

There is another option, perhaps the most advanced one yet when it comes to completely anonymous Internet surfing. That option is Tor. Originally developed by the US Government, Tor is a type of "onion router" that routes your internet traffic through a complicated layered system. There is much to say about Tor and a lot of explaining behind how it works. If you are interested in it, you can visit the Tor Project [on its website](#).

If you would like to use Tor for anonymous browsing, it's easy to do so. However we will not be installing Tor using the Ubuntu package repository, like has been done in the past. Since Tor updates are considered very important for stability and security reasons, we want to make sure that we are getting them on time. For this, we will patch Tor's custom update server into our Ubuntu installation. That way, whenever we run `sudo apt-get update` and `sudo apt-get upgrade`, Tor will update itself whenever a new version is available.

First, run `cat /etc/debian_version` to check your Ubuntu's version codename. If you are using 12.04, the codename is "precise." Next, open up `/etc/apt/sources.list` and add the following line, with your version codename in the appropriate place:

```
deb      http://deb.torproject.org/torproject.org $codename main
```

Next, add the Tor project's GPG key, used to sign its packages and verify their authenticity:

```
gpg --keyserver keys.gnupg.net --recv 886DDD89
gpg --export A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89 | sudo
apt-key add -
```

Then the final few commands:

```
sudo apt-get update
sudo apt-get install deb.torproject.org-keyring
sudo apt-get install tor
```

From this point on, Tor is installed and running on your system. But before you can use it, you must configure your browser to use it. You can do this manually of course, but we will use the most convenient and automatic method -- via a browser plugin. Download the [Tor Browser Bundle found here](#). Make sure you download the Linux version, and the

architecture that corresponds to your computer. If you don't know your architecture, run `uname -m`. If you get "x86\_64" as a response, you have a 64-bit system; if you get "i386" or "i686" as a response, you are using a 32-bit system.

After downloading the package, run the following to extract it and install:

```
tar -xvzf tor-browser-gnu-linux-*.tar.gz
cd tor-browser_*
./start-tor-browser
```

This will start a specially-patched version of Firefox that has Tor enabled. You can create a shortcut to the `start-tor-browser` script on your desktop or in the sidebar, and you will be able to launch your Tor browser whenever you want. You will need to reinstall your Add-ons in this Tor browser, and you will not be able to use your old browser (Chrome or Firefox) if you want to have the protection of Tor. However the Tor browser is based on Firefox, so any plugins that work for Firefox should also work for the Tor browser.

Before you start using Tor, there are some things you should be aware of before you start surfing! Make sure you [check out the list](#) and are aware of what they might mean for you.

## 2.4.2 - Secure Your Email

### Encrypt Your Connections With SSL/TLS

Just as it is important to use websites that enable SSL, you will want to do the same with your email connection. If you always use your email in a browser, like Yahoo Mail or Gmail, you don't need to worry about this. But if you use a third-party client like Thunderbird, there are settings you should make sure are set.

In Thunderbird, click Edit > Account Settings. On the left side of the window, you will see an expanded list of email accounts. Choose the one you want to set for SSL and click "Server Settings." Connection Security should be set to STARTTLS. Then click "OK." If you choose this and your email does not send or receive, select SSL/TLS in this field instead and try again

If you experience complications enabling SSL in your email client, your email provider will give you instructions on how to do so in its Help section. It may have a different server name or port configuration for you to enter here.

## Encrypt Your Messages with PGP

PGP is the standard for email encryption nowadays. It allows you to seamlessly encrypt mail messages to people and have them just as easily decrypt them upon receipt. You might send a full message to someone, and if anyone that might come across your message happens to open it without your key, this is all they will see:

```
-----BEGIN PGP MESSAGE-----
```

```
Charset: ISO-8859-1
```

```
Version: GnuPG v2.0.19 (GNU/Linux)
```

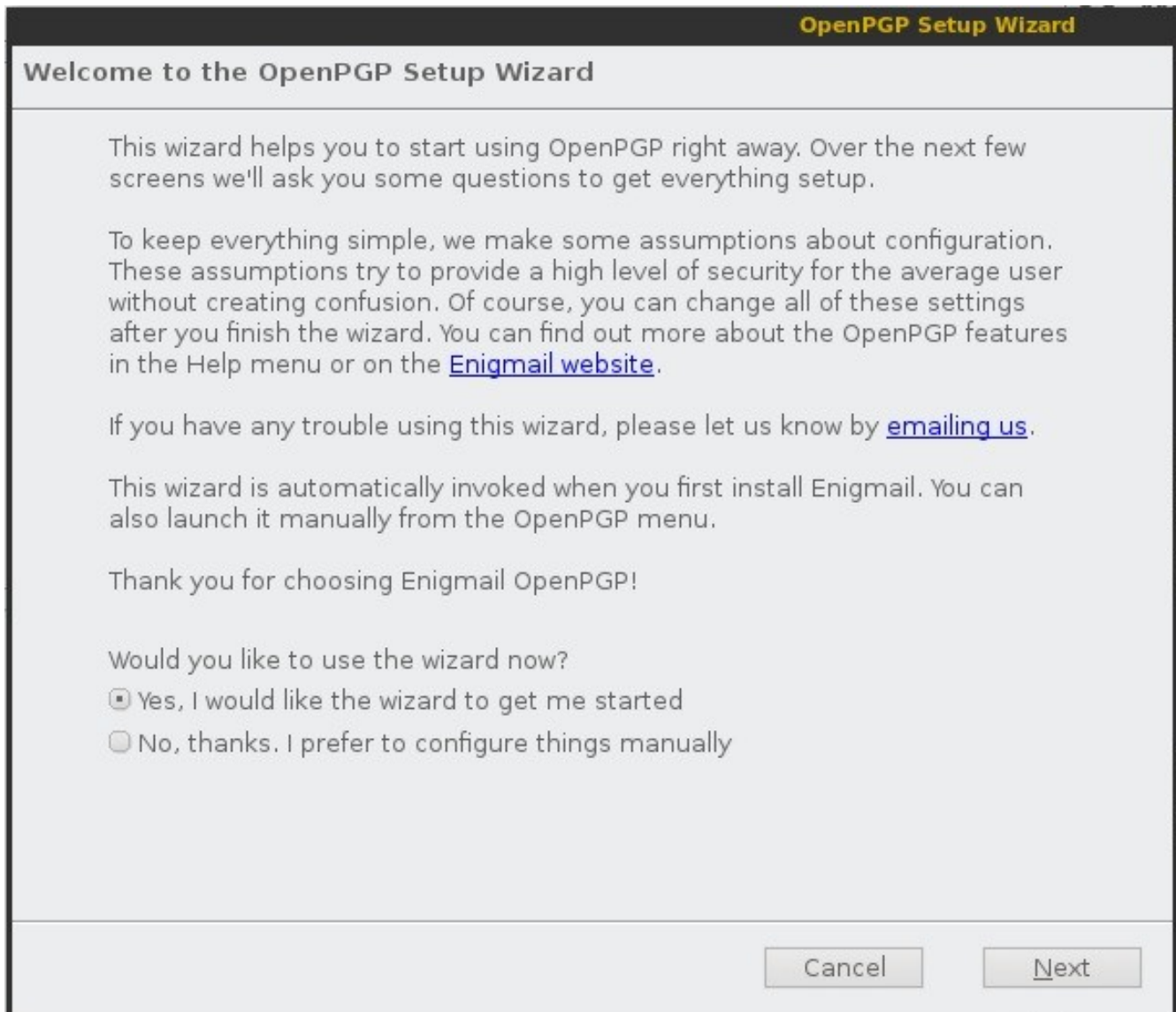
```
hQEMAYL1sE8aLy0uAQf9G12ng+ijfKmMEyInN6iauYaP6ITIrzOtTK+ZiEHDc1oAeKZwh
4zgt1O6l11AUU+nYC1WCTMKP1cIU0yOqp1INF19ZNtn7qNneUcmYmfyaDBATpz15qXiM5
mVMCrK82e1XtGLRkko76In4oh8WEVxISZhw4ATD+Vx0jXqQR6HUkeK1sr4a+OTjSZlT+i
TYy0Q2PQjSLMp5xKyjoYt9ArxOQDBbznwRcwfRIMzUnCf2Q87uayssbp5HmnpZj8Izgm7
/EehrkQfnklhAvgGPrNk/d8oD+PK2D4h3plAqpSres607k6OAehppAJt/TKUYoNZeM6qC
eBOrRQohuSmGg3kNNLpAUJOONXIYFavuc2Iyb+phyBRsXrcZJ/e2PN/Xx7i6Ki/P3347f
oZ0/GaVpUrWP9MQJLjawR/cVEEBY2lar4t...
```

An indecipherable mess is all that awaits them.

For PGP to function properly, you must generate a "keypair" for yourself, and you must have the public key for your chosen recipient. Let's go through the steps.

We will use Thunderbird and it's Enigmail plugin to handle our email encryption and decryption. In Thunderbird, click Tools > Add-ons, click Get Add-ons, then search for and install Enigmail. Once the plugin is installed and Thunderbird has restarted, click the OpenPGP menu, then choose the Setup Wizard.





Click Next, then choose the email accounts you want to use encryption with. (Remember that you will have the choice whether or not to encrypt each message, so you don't have to worry about making everyone you know get PGP keys if you don't want to encrypt your emails to them!)

Click Next again, and follow the rest of the wizard. It explains well the steps and options you need to choose, and it also helps you automatically generate a PGP key.

Now, once this is complete, you have the option of submitting your public key to a keyserver. A keyserver is like a search engine for people's public keys -- if you have someone you wish to communicate with, you can import their key from a public keyserver without them needing to give you their key directly. This does not reduce the security of your keys, as the message can only be decrypted by the specific recipient anyway. You are not required to upload your public key to a keyserver; if you choose not to, you will need to keep your messages signed with your PGP signature (which Enigmail usually enables by default), or you will need to export a copy of your public key to an .asc file and give that to your conversation partner.

To upload your key to a public keyserver with Thunderbird, click OpenPGP > Key Management, then right-click your key and choose Upload Public Keys to Keyserver. It doesn't matter which server you choose at this stage, as they all will share their data with each other.

To find someone else's public key on a keyserver, open up Key Management then click Keyserver > Search for Keys. Type in the email address of the person you want to email, then check the box next to their name. If their name doesn't come up in the list, you can import a public key that they give you in .asc format by choosing File > Import Keys From File. It is usually a best practice to use a key that is given to you from someone rather than using a public keyserver, if you trust them.

Remember that if you have uploaded your public key to a keyserver, you are pretty much locked into using that key. If you ever lose your keyfiles or want to change keys for some reason, you will need to generate and upload a revocation certificate. This is done to ensure trust, and the knowledge that the keyholder really is who he purports to be via their name and email address.

After this, you can write encrypted emails to whoever you want, provided you have imported their public key! If you chose to automatically encrypt your messages in the Setup Wizard, you don't have to set anything; if not, you can click OpenPGP > Encrypt Message in the New Message window to write an encrypted message. Once you click "send" and enter your password, the message will automatically be encrypted. When you receive an encrypted message from a conversation partner, Thunderbird will automatically ask you for your password, and will decrypt the message for your viewing. The message will remain encrypted so you will need to enter your password each time you wish to read it.

### 2.4.3 - Secure Your Chat Applications

#### Encrypt Pidgin Chats with OTR

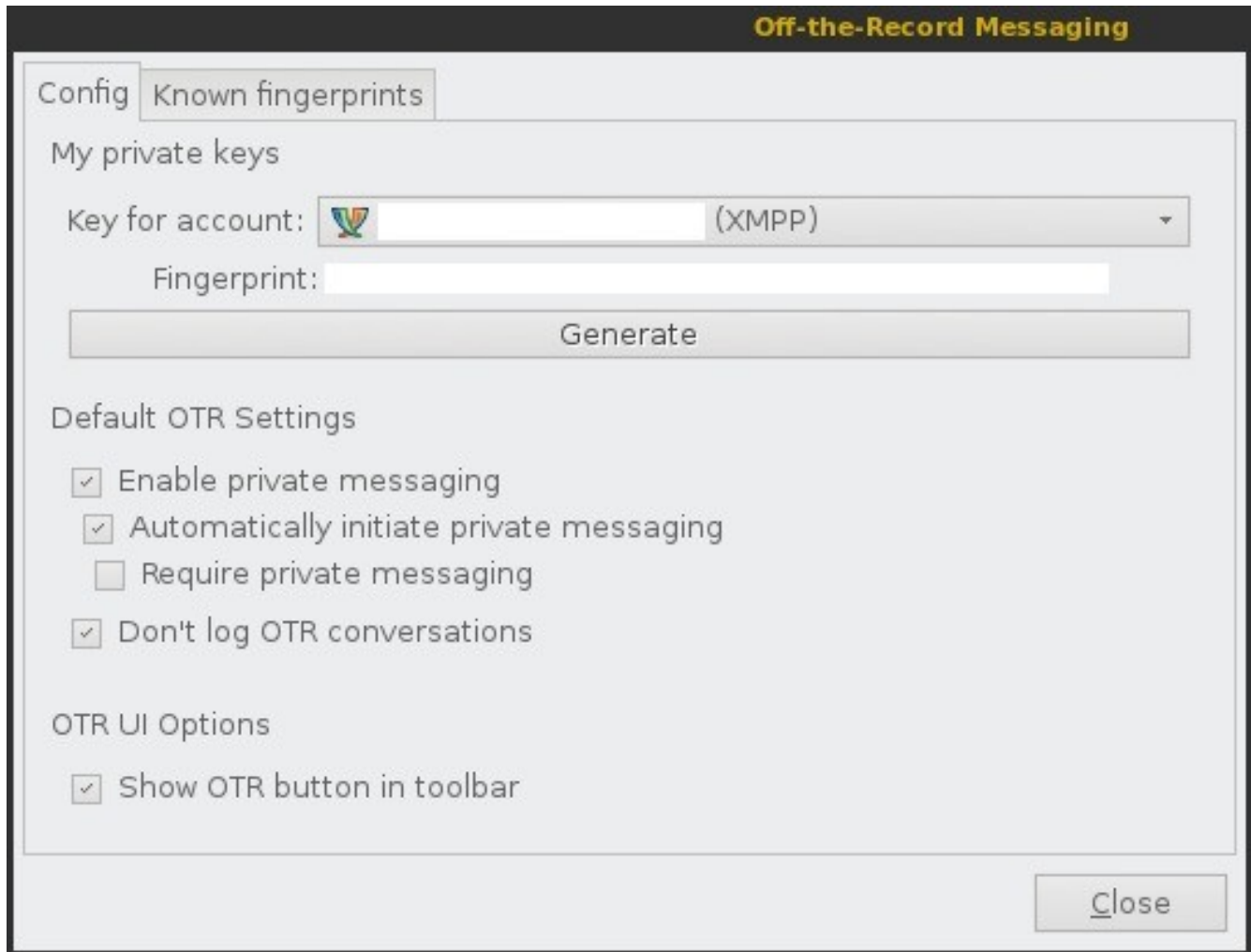
If mail is a bit too slow for your taste and you prefer Instant Messaging (IM), there is a solution for you. The chat application Pidgin, a mainstay of Linux communication suites, has a plugin named "OTR" (Off The Record) that can be used to encrypt your chat conversations. It operates in a similar way to PGP, in that you must first exchange public keys with your conversation partner. If you don't already use Pidgin, it is available for install in the Ubuntu repositories.

To install the OTR plugin, head to [the Cypherpunks site](#) and download the tarball for the OTR Library and Toolkit, as well as the one for "OTR Plugin for Pidgin." Then run the following:

```
tar xzf libotr-*.tar.gz
cd libotr-*
./configure --prefix=/usr
make
sudo make install
tar xzf pidgin-otr-*.tar.gz
cd pidgin-otr-*
./configure --prefix=/usr
make
sudo make install
```

This will install both the required libraries for OTR as well as the plugin specific to Pidgin.

To configure the plugin, open Pidgin and click Tools > Plugins. Check the box next to "Off The Record Messaging." Then, click the entry for "Off The Record Messaging" and choose Configure Plugin.



Here you can choose a set of options based on how you want the plugin to behave. Also, you can choose to generate a key for a specific account. Once you begin a conversation with a friend who also has OTR enabled, you will see a notification display that you can begin a conversation with that person. Click "Not Private" and choose "Start Private Conversation" to enable encryption with the active conversation partner. And you're off! OTR is notoriously easy to set up and use.

### **2.4.4 - Further Reading**

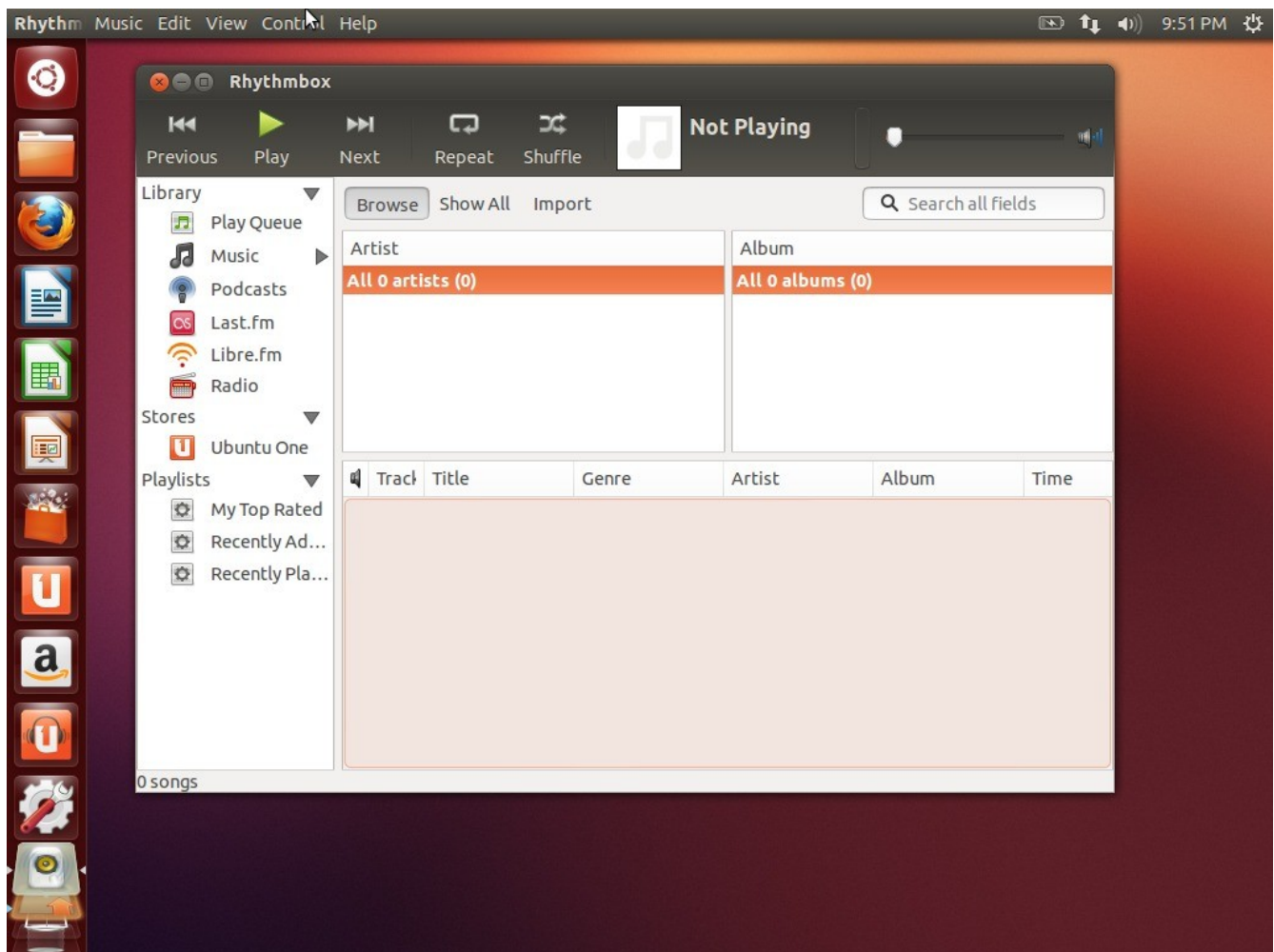
- [How To: Protect Your Privacy with Ghostery - Chip.eu](#)
- [Tor documentation for Linux](#)
- [Enigmail PGP Quick Start Guide](#)
- [How to Use OTR to Initiate a Secure Messaging Session in Pidgin - Tactical Technology Collective](#)

## 2.5. APPENDIX: Popular Applications

The following is a non-exhaustive list of frequently used applications and file formats that may make your switch to Linux easier. There will be multiple choices for some types of applications. On Ubuntu, most of these applications can be found in the Ubuntu Software Centre, or by running `sudo apt-get install $appname` in the Terminal.

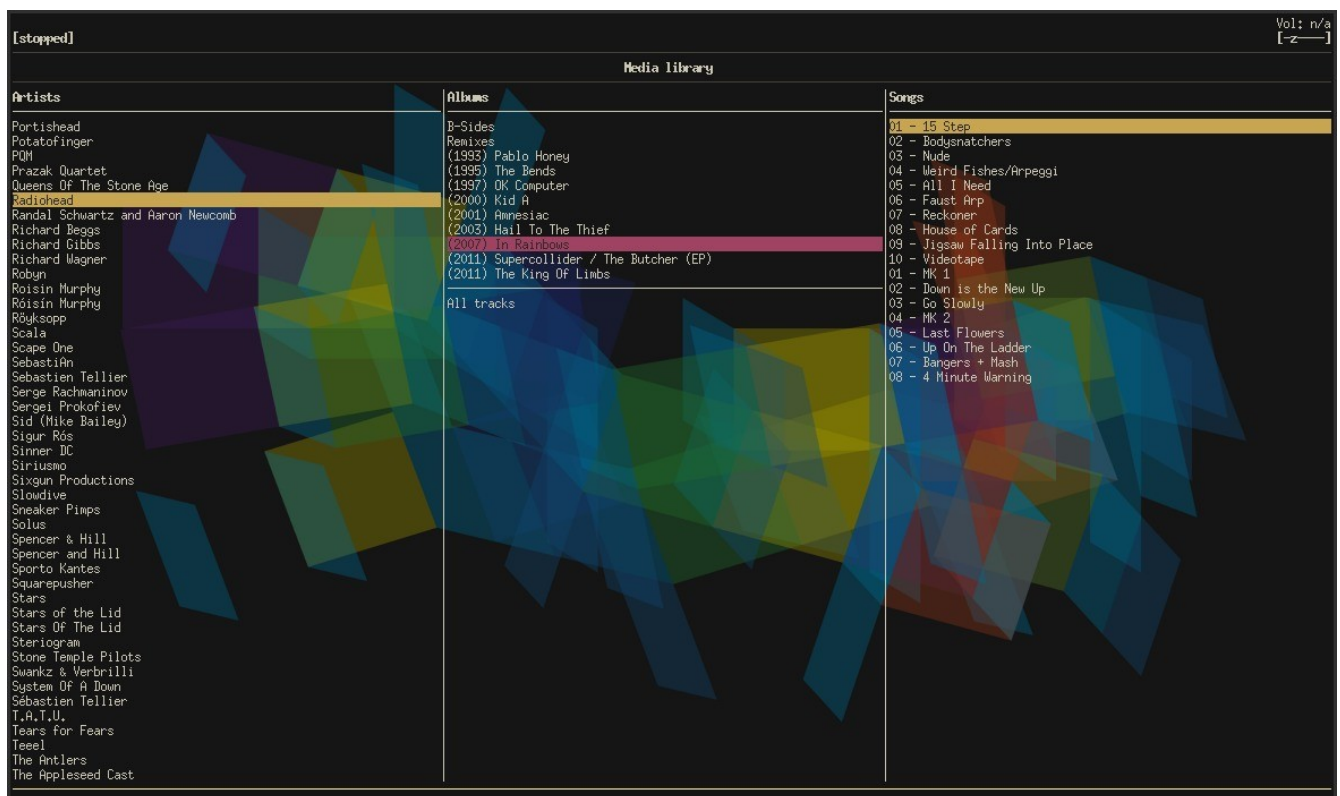
### 2.5.1 – Applications: Media

The default music player that comes with Ubuntu is **Rhythmbox**. Rhythmbox is a decent music player with many features similar to iTunes. It has an easy-to-use library view, with integrated podcast, Last.fm and music store integration. It also features a plugins system that can extend its use beyond simple music playback.

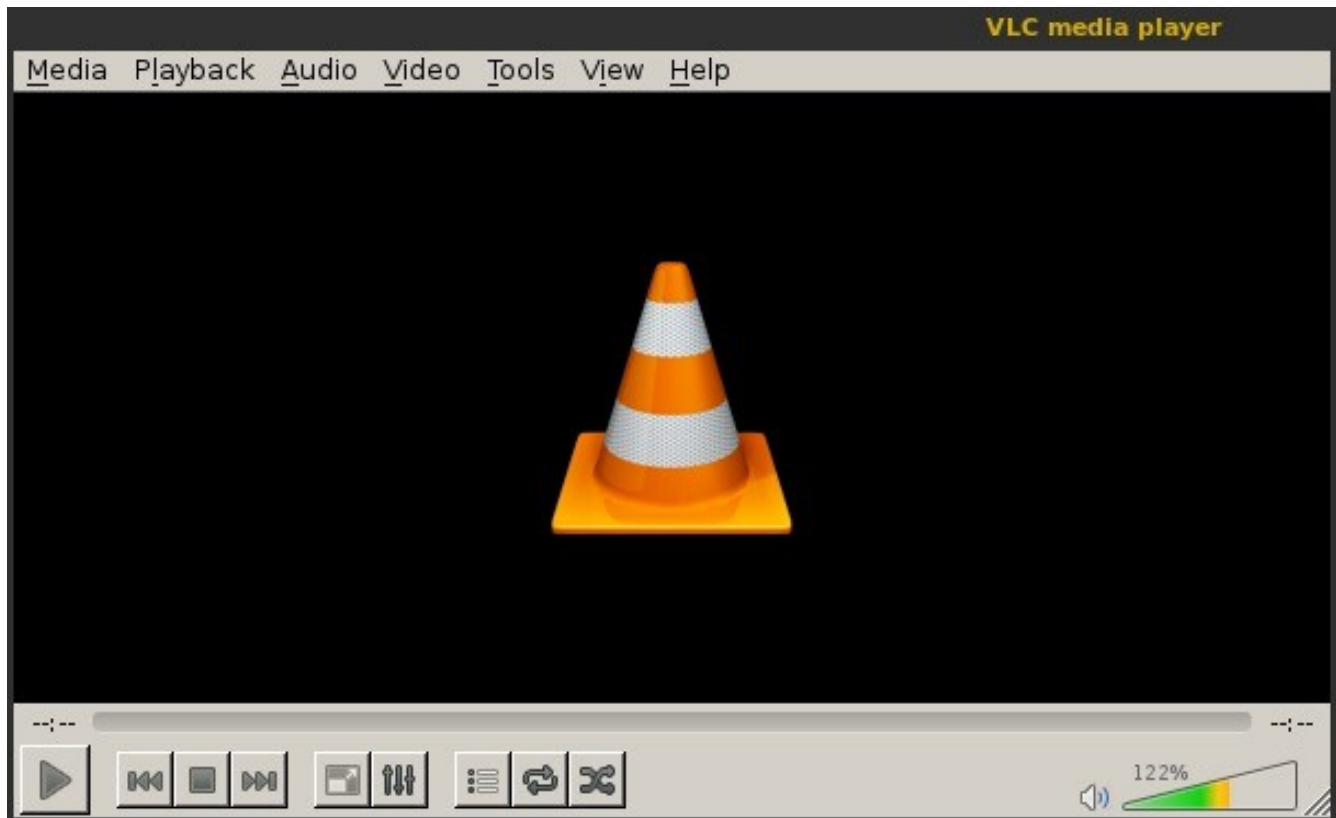


**Banshee** is also a good option, and it is even *\_more\_* like iTunes for those who are used to its interface. For those who use KDE, you can check out **Amarok** or **Clementine**.

Another option for more advanced users is **mpd**. Mpd is technically an audio server that streams to local clients. When you use mpd, you will therefore set up the audio server (which is always running) as well as a client to interface with it. A favourite mpd client is **ncmpcpp**. It has a strange acronym of a name, but it is very fast and has a fully functional graphic/command-line interface. For those who like to customize their desktop environments, ncmpcpp is a hit, as it is as customizable as any other Terminal window.



The old standby for playing video on Linux is **VLC**, much like it is on other platforms. It can play a very wide variety of different video formats, supports subtitles and multiple audio tracks, and is also extensible by plugin. It's also very fast!



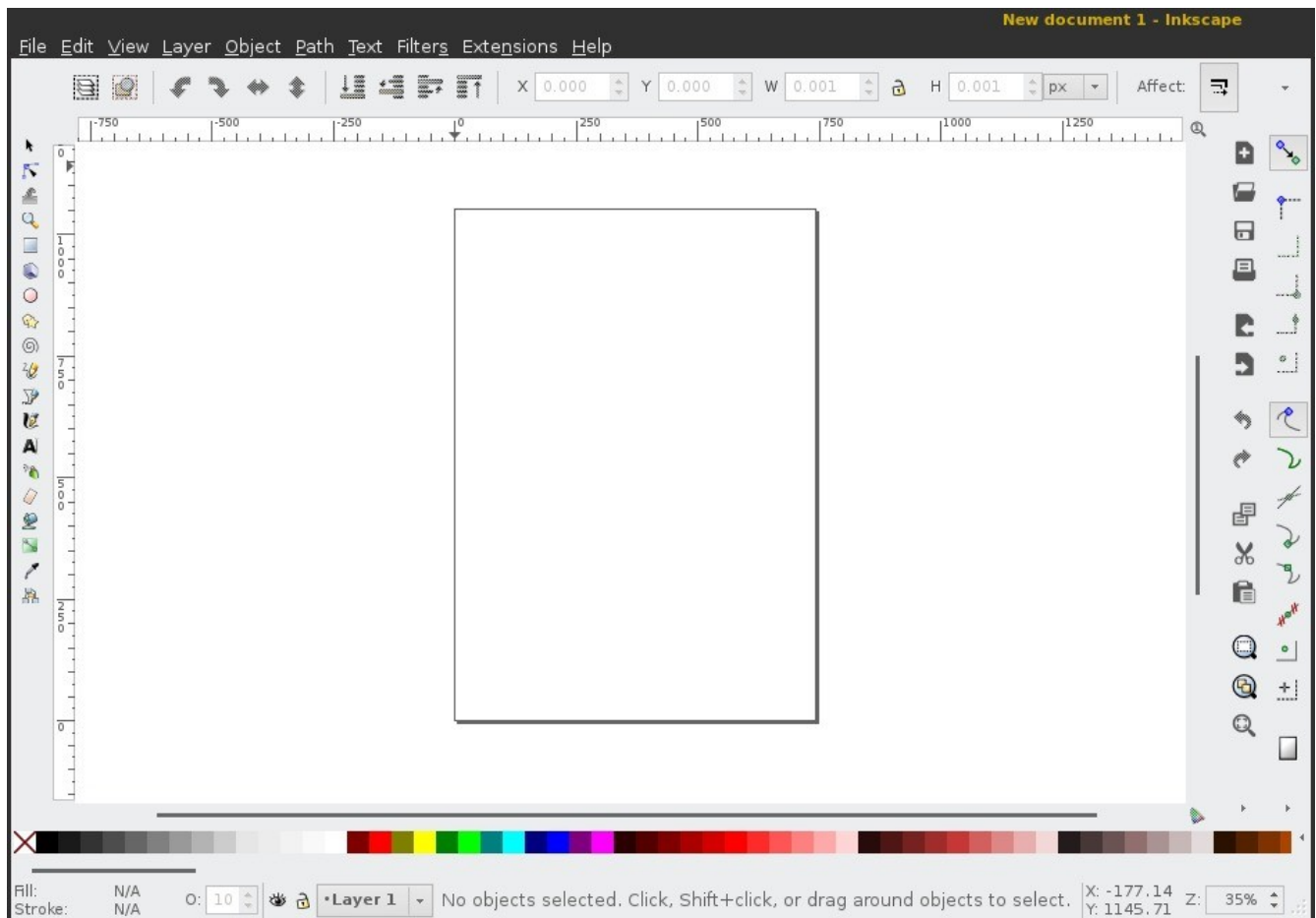
Ubuntu comes with a standard image viewer called **Image Viewer**. This is analogous to Windows' Image Preview, bringing decent quality image viewing to the GNOME desktop. For other desktop environments or distributions, **Viewnior** is a very fast and lightweight replacement for Image Viewer and is highly recommended.

Keeping photo libraries on Linux is easy with **Shotwell**. Shotwell is essentially a Linux clone of the popular iPhoto for Mac OS X. You can import images from your hard drive or directly from your digital camera. Archive your photos by date, by event or by tag.

For editing graphics, the most common open source solution is The **GIMP**. While not quite as fast or as usable as Photoshop, The GIMP is still very powerful and actively developed, bringing intensive image manipulation capability to Linux.



If you work with vector images or graphic design on a regular basis, check out **Inkscape**, which has many of the same features as Adobe's Illustrator.



The most-used option for audio editing on Linux is **Audacious**. Audacious is also widely used on other platforms like Windows. It is easy enough to use for beginners to audio editing or podcasting, but flexible enough for experienced professionals.

For webcams, **Cheese** is a good option for GNOME-based desktops. Fans of ebooks and keeping digital libraries can check out **calibre**, which is a very powerful and feature-rich ebook library. **Brasero** comes default with Ubuntu, and is used for CD/DVD burning.

## 2.5.2 – Applications: Utilities

Ubuntu's default text editor is **gedit**. Gedit is a fine standalone text editor for infrequent use. Another very fast and lightweight option is **leafpad**. For more text editors that might be of better use while programming, check out the Productivity section.

Ubuntu comes with a standard archive manager called (wait for it) **Archive Manager**. From here, you can easily create or modify your archives of many different types.

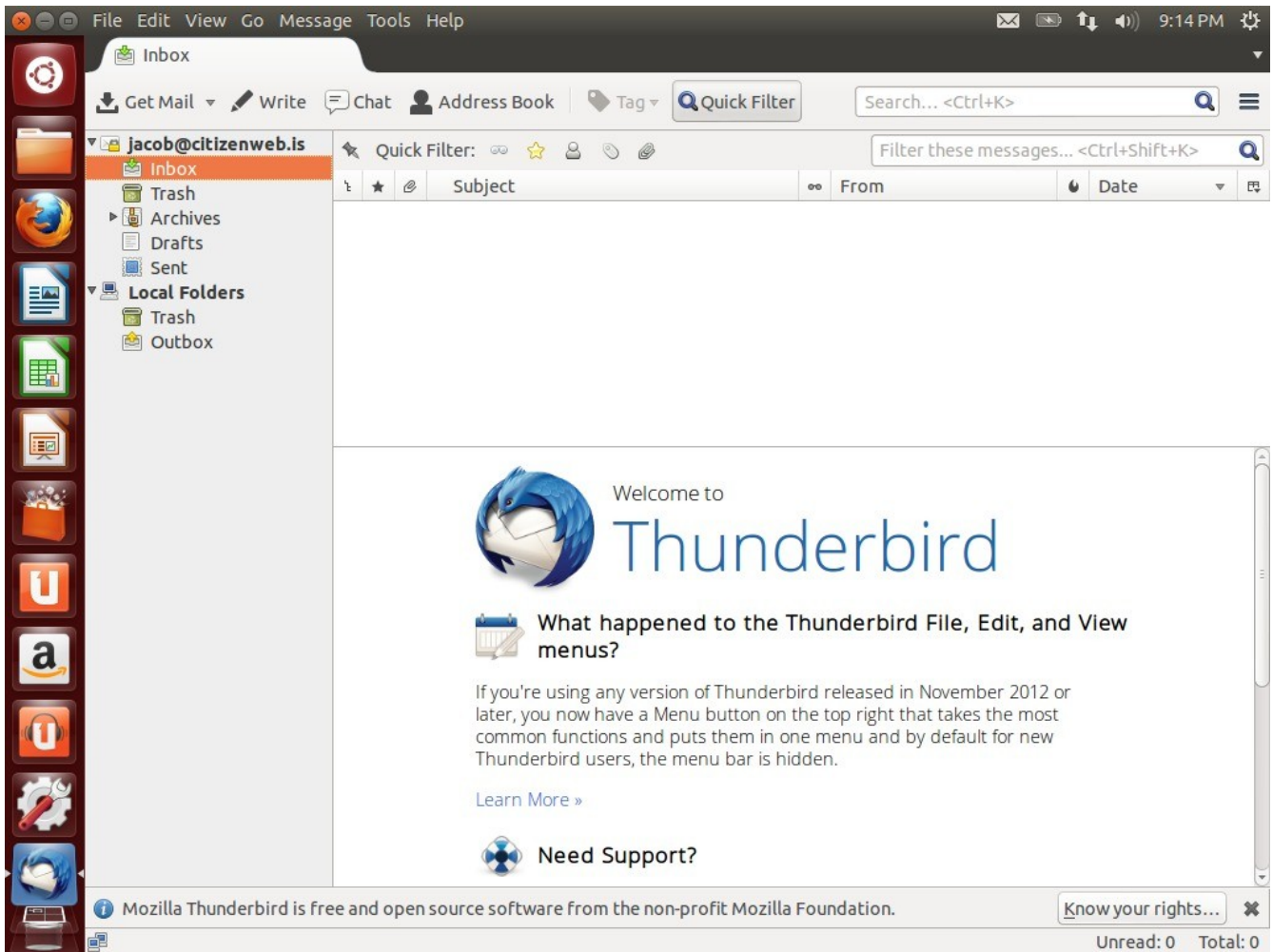
**TrueCrypt** is very often used by those who work with sensitive files, or simply wish to encrypt/password-protect some folders on their system.

Other utilities of use include the **Terminal** for running commands, or **vinagre** for VNC connections to other computers.

## 2.5.3 – Applications: Networking

Ubuntu comes installed by default with **Firefox**, the common cross-platform browser that (nearly) everyone loves. If you don't love Firefox, you can install **Chromium**, which is the Linux version of Google Chrome. There is also **Opera** or other browsers available for Linux.

For email, the main choice is **Thunderbird**, which is also installed by default in Ubuntu. It is analogous to Mail in Mac OS X, or to Microsoft Outlook for Windows. **Evolution** is the runner-up in the Mail category, which is included by default in the GNOME desktop. **KMail** is a decent option for KDE users.

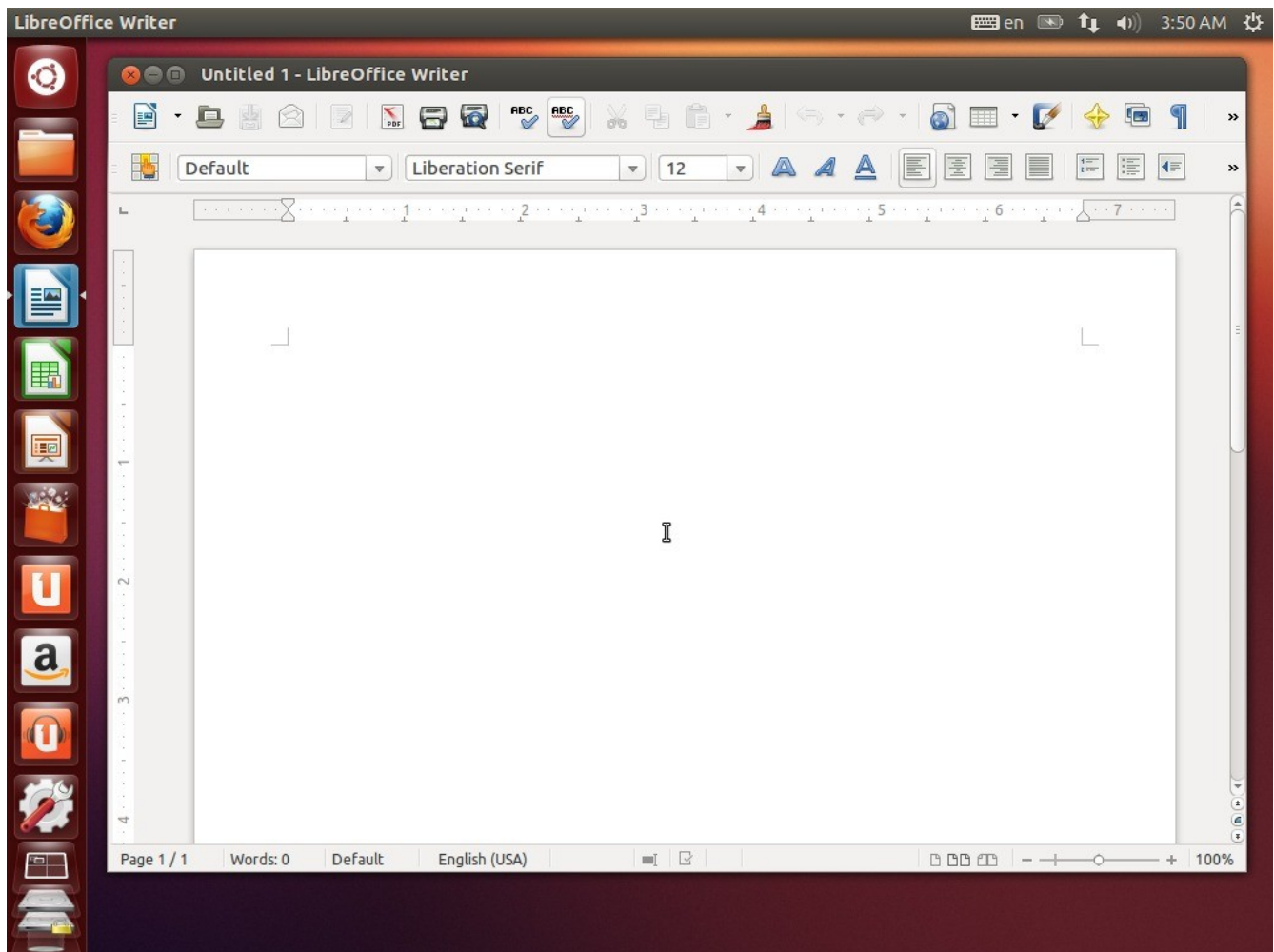


For instant messaging, **Pidgin** is commonly used. You can use Pidgin with AIM, ICQ, MSN/Skype, Google Talk, XMPP, Facebook, IRC and many many other protocols. It is easy to use, and supports a wide variety of plugins to extend and personalize its use. **Empathy** is the client that comes built-in with Ubuntu, and it supports a great deal of protocols as well. Other choices include **irssi** for a command-line IRC client, or **Quassel** for a full-featured deluxe GUI IRC client.

If you are a frequent microblogger from your desktop, **Qwibber** comes built in with Ubuntu, and supports posting to Twitter and Identica. Other than that, **Polly** is a fantastic standalone Twitter client for the GNOME/Unity desktop. Those who read RSS feeds from desktop applications can check out Liferea, RSSOwl, or Akregator (KDE). Finally, **Transmission** is frequently used for torrent downloads and management.

### 2.5.4 – Applications: Productivity

The king of open source productivity software on Linux is presently the **LibreOffice** suite. LibreOffice includes a word processor, spreadsheet editor, presentation creator, math formula creator, and simple graphic design program. An alternative to LibreOffice is the **OpenOffice** suite, the ancestor project to LibreOffice.



Ubuntu comes with a built-in PDF reader called **Document Viewer**. It can view and edit PDFs as well as other document formats like PostScript. Lighter options for PDF readers include **Zathura** or **MuPDF**.

For programming text editors, **Geany** is a good option. Other options include **SciTE**, **Bluefish** or **Scribes**. If you are looking for a more full-featured IDE, you can try **Eclipse** or **Aptana**.

## ***The CitizenWeb Guides – Your Personal Server***

### **3.1. Why a Personal Server?**

---

The short answer is: **Because you don't have to sacrifice features, functionality or comfort just because you are concerned with security and privacy.**

#### **3.1.1 - The Pros**

Many people look to Google, Facebook and other large platform services for the exceptional convenience they offer. With all of the services available to us online these days, it's easy to see how they can improve our lives and make us live or work better. However there are significant risks to using these services; risks that are only deepening and becoming more serious with time. What most people do not realize is that, once the initial investment of buying or hosting your personal server is passed, self-hosting data is very easy and requires little to no sacrifice of functionality.

Are you addicted to Google Calendar and can't live without it syncing across your computers and devices? Check out ownCloud, which lets you do the exact same things, but gives you the control over your data that Google can no longer provide you with. Are you lost without your Gmail account? You can host your own email and have all of Gmail's features in the client of your choice. Plus, you can still sync your mail and contacts effortlessly across your devices.

You can have your own "personal cloud," a customizable platform service that meets your needs, without selling your personal information to marketing agencies or overzealous governments. You can do it by hosting your very own Internet-connected server.

The most substantial "pro" to hosting your own data with a personal server is the privacy factor. As mentioned repeatedly in this guide, data given to platform services like Google or Facebook risks being handed to marketing agencies or governments without your consent, and in some cases without you even knowing. When your data is self-hosted and properly secured, you can be sure that your information will not fall into the hands of marketers. Furthermore, governments will be required to physically intervene with warrants or other methods if they suspect you of something, which is much less common and costly than the bulk interception they practice today.

For these reasons, self-hosting your own server is a huge plus for activists, whistleblowers or journalists. But it is also very important for common, everyday Internet users like you and me. The more data we share about ourselves online, the larger that Google and Facebook get, the more irresistible targets they will make for marketers and governments. We are already seeing today how simply standing up for what is right in society can get you bullied, threatened, abused, extradited and worse. If you are sure that nothing you do right now can get you into trouble, can you be sure that in ten years from now, the positions you take or the data you own **\*\*now\*\*** won't be used to get you into trouble? The Internet is a time machine -- any comment you make on a platform service can be indexed and potentially used against you. This is why a default state of privacy must be enforced on the web -- and if services like Google or Facebook won't do it for us, then we must be prepared to take matters into our own hands, by self-hosting our data and refusing to participate in their systems.

#### **3.1.2 - The Cons**

Decentralizing the Internet isn't always a field of flowers -- sometimes it can be a downright annoying experience. There are a few different pitfalls that one must be aware of before they take the plunge and host their own server.

Perhaps the most significant drawback is in downtime. Google's services, while they have been subject to very public and unexpected downtimes in the past, are overall very stable and well-managed. This cannot possibly be matched in a home server environment, when data is isolated to only one node. If you host your server at home, this server will be subject to any power outages, Internet service interruptions, or accidental unplugs when your cat tries to make a home behind your computer. Once a downtime occurs, you will not be able to interact with users; i.e. people will not be able to see your web server, send you emails, or do much of anything else.

Next comes the security aspect. Every server on the Internet represents a target for hackers and script kiddies. Once they can get access to a vulnerable machine, they can try to troll through it for your personal info, or just use it as a host for spam mail or monitoring your Internet use. You will not have the security experts at Google making sure that your services are under lock and key -- **\*\*you\*\*** will be your own security expert. Luckily this is not very difficult, as the tips outlined in this guide should diffuse a decent majority of common attack vectors. However nothing is 100% secure, and a self-hoster must remain vigilant that their configuration is frequently updated and not compromised.

Because of these downsides, contingency plans should be made often. If you have the resources, rent a VPS that you can switch to if your main server goes down. Practice frequent encrypted backups to external media or offsite locations. Make sure to reduce your risk of "going down" as much as possible if you are going to be hosting critical content.

### **3.1.3 - Types of Servers**

If you don't have the space to set up a traditional dedicated server in your own home, or are unable to do so for other reasons, don't worry -- there are a few different ways to self-host your data, and we will look at each of them here.

#### **Dedicated Server**

This option consists of having a standard computer in your home that is connected to the Internet and/or a home network. This server can be any used desktop computer that you have lying around, or a custom-built one from ordered parts. Once the computer is ready, it can be stored in a closet or a tucked-away corner of your home. It does not require a constant monitor or keyboard/mouse connection to be functional; you can communicate with it via SSH (explained in this guide) to configure or maintain your running services.

This option is the best for running a large amount of online services at once. As it has more processing power than embedded miniservers, it can handle more services and more visitors than a Raspberry Pi might be able to. Also, while it is more expensive from the start (reasonable cost estimates for a brand-new dedicated server run between \$500 and \$900 US dollars), a dedicated server can be more cost-effective in the long run when compared to the monthly cost of a virtual private server (VPS).

However, as suggested above, dedicated servers do take up much more space than embedded miniservers or (obviously) VPSes. They require a larger initial investment, and will generally require special services from your Internet Service Provider (ISP) in order to make them fully functional. Also, in case of a move, power outage or other unforeseen service interruption at your home, you will be without a way to host your content until the interruption passes.

## **Embedded Miniserver (Raspberry Pi)**

This is a relatively new option when it comes to self-hosted servers, but it is one that is rapidly gaining popularity. Raspberry Pi minicomputers can be purchased for only \$25USD. With an exterior case and a dedicated network connection, they can offer a host of simple server applications, such as web servers, email servers and databases. These miniservers cannot be beat when it comes to the initial investment cost, providing a huge advantage to those who do not have hundreds of dollars lying around. They also still provide the security of physical ownership and constant access that a VPS cannot offer.

Embedded miniservers are, however, decidedly slower and not able to handle nearly as much load as a dedicated server box. Their use should be restricted to offering simple web services only, and not heavy media-intensive server apps. And as mentioned above, these servers are still hosted at your home, so they will still be subject to occasional power outages or other interruptions as they affect you.

## **Virtual Private Server (VPS)**

A virtual private server (VPS) is a virtual machine that is hosted elsewhere. This is done typically by a hosting company. The difference between VPS hosting and traditional web hosting is that you can run anything on a VPS just as if you were using your own physical computer. You can access your virtual server via SSH or VLC from wherever it is in the world.

VPSes have many benefits over other server types. First, they do not require a massive initial investment, like a traditional server might. They are usually offered for monthly or yearly fees paid to the hosting company. As the server is virtual and hosted elsewhere, you do not need to worry about storing it in your home, nor do you need to change your account with your ISP. Furthermore, if you are a whistleblower or activist and live in a country with particularly egregious monitoring or seizure laws, you can order a VPS in a country that does not have such stringent rules. For example, there are VPS companies in Iceland, a country known for its freedom of speech and protections for journalistic publication.

These virtual servers do have their downsides. First, they are generally not quite as capable as dedicated home servers, but are still better than embedded miniservers like the Raspberry Pi. You can purchase a very powerful VPS, but this will likely cost you a significant monthly fee over the standard package costs. This leads to the second point: the aggregated cost that you pay for a VPS over many months will undoubtedly be more than what you pay for just buying a dedicated server. And finally, there is always the issue of personal assurance: you cannot physically assure the security of data on your VPS. The VPS may also be subject to the snooping or seizure laws of the country it is based in, regardless of your own nationality. It is often a good idea to encrypt any personal data stored on a VPS because of this.



## 3.2. Before You Begin: Options, Configuration and Hardware

---

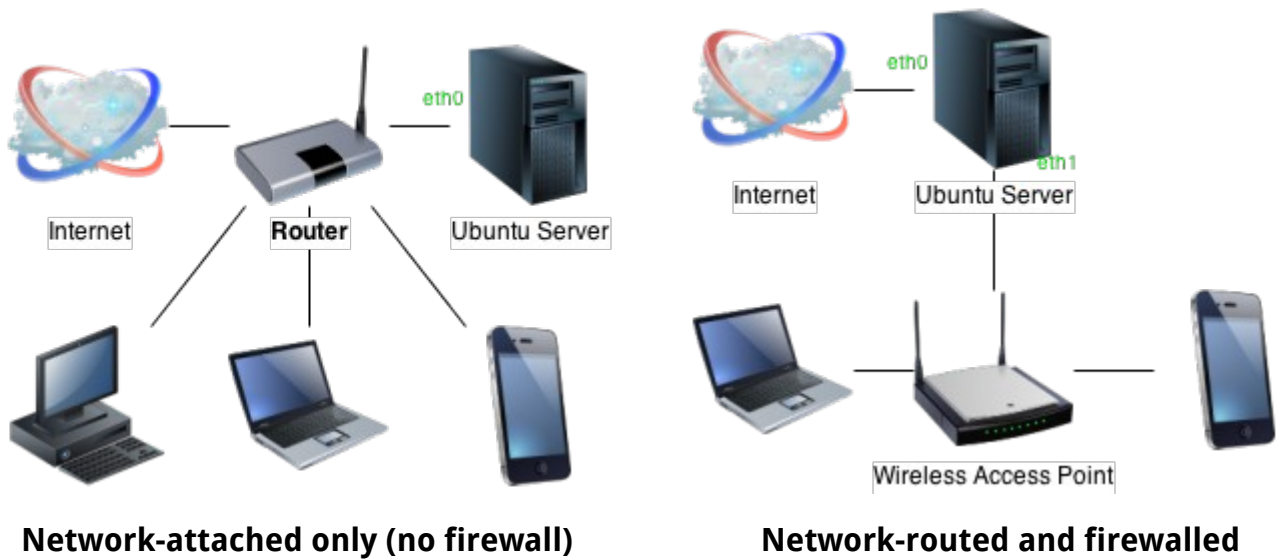


(Note that Virtual Private Server (VPS) users can skip this article entirely. Embedded miniserver users like those with the Raspberry Pi can skip down to section 3.2.3.)

### 3.2.1 - List Your Options

What do you want your server to do? What will it be handling for you on a daily basis? These are important questions to answer before shopping for your server hardware.

- **Will you be running hardware-intensive services?** Servers that run virtual machines or media servers traditionally have much higher hardware requirements than simple email/web servers. The more VMs you want to run, the better CPU you will need; similarly, the more media services you wish to host, the more memory you will need. For any server that is to run a media service, it is recommended to have at least 8GB of memory.
- **Will this be a "headless" server?** Will this computer need to be used directly, or can you simply put it in a corner and manage it from your laptop via an SSH connection? If you need to access it more than once, it would be a good idea to buy a monitor as well. Keep in mind that you will need to use a monitor during the server setup, but if you have another desktop, you can borrow that monitor just for the installation.
- **Will this be a firewall or network controller?** Do you plan on using this computer to serve as a firewall instead of using your existing router configuration? Will this computer be serving DHCP clients, or will you leave that to another router connected to the network? If you've answered yes to any of those questions, it would be a good idea to get a server motherboard equipped with two ethernet ports (NICs). One will be "front-facing," that is, connected to your cable/DSL modem; the other will connect to a hub or wireless access point for your internal network.



### 3.2.2 - Buy Hardware

Now we get to the fun part - doing some shopping! Load up your favourite computer parts vendor and let's get started.



Popular parts vendors in the US and Canada are [Newegg](#) and [TigerDirect](#). Newegg usually has the better prices and availability, but whichever one you pick is up to you. It's usually best to make lists on a few different sites to see which one actually has the cheapest price for that specific application. In the UK/Europe, check out [Misco](#).

## CPU

The most popular server CPUs these days are Intel, hands down. The Sandy Bridge and Ivy Bridge-class processors are really without comparison when it comes to performance and dependability. You can find decent ones for between \$250 and \$350 that will provide more than enough power for what we are looking to accomplish with our server.

It is also important to remember your CPU's cooling requirements. Most new Intel CPUs come with cheap but decent cooling fans; though if you are looking to improve your server's noise production, it may be a good idea to buy a nicer fan as well. Just make sure the fan is compatible with your chosen CPU's socket type.

## Memory

Some individuals and companies may consider this heresy, but you really don't need to buy the most expensive RAM out there in order to have a dependable and quick system. If you are spending more than \$150 on RAM, you are very likely spending too much. Decent server memory is not too much more than normal memory.

## Motherboard

The motherboard is where the entire system comes together. Choosing one depends on the services you wish to offer with this server.

99% of the time, you will want to choose a server motherboard. These boards support server-class CPUs like the Intel Xeon series. Furthermore, most of them come with two Ethernet ports (NICs). This is indispensable for servers that act as routers for internal networks, or servers that will host email/web services. A common setup is to plug the cable/DSL modem into the first NIC as a "front-facing" interface, then to route the internet connection through to the second NIC, which is connected directly to your network hub or wireless access point.

It is possible to get by with a standard motherboard and CPU if you only want to do media sharing on your internal network, but if you are even *\*considering\** doing more than that, it's best to go for the server motherboard and CPU.

Regardless of the class of motherboard you go with, the most important match you will make is between motherboard and CPU. You **MUST** remember to pair them by their socket type. For example, socket LGA1155 CPUs might not fit every socket LGA1366 or LGA2011 motherboard, etc.

Also keep RAM (memory) in mind. Motherboards have different types, sockets and speeds for RAM, as well as limits to how much memory they can handle, so make sure you can find one that works with your memory requirements. Your motherboard's manual, usually available in PDF from the manufacturer's website, will have all of this information.

## Case

Cases might not seem like an important consideration, but there are two critical elements to be aware of when choosing one to meet your needs.

- **Size:** There are many size designations for motherboards: ATX, Mini ATX, Micro ATX, etc etc. Make sure the case is the correct size for the motherboard you are looking to purchase.
- **Power Supply:** Most cases these days come with their own power supplies, but they are not all created equal. If you are planning on purchasing a computer with an Intel server CPU, you will definitely need a power supply with 24-pins (or "20+4"). The extra 4 pins are required to meet the motherboard and CPUs extra requirements. Keep in mind that, if you have your heart set on a particular case that comes with an incompatible power supply, you can always remove the old one and install one separately purchased.

## Hard Drive(s)

Again, the type of hard drives you will need will vary depending on what you want to accomplish with them. For simple web/email servers, you will not need much space at all. For those looking to do any sort of file hosting, space will likely be very important. You can pick a certain number of drives that can be matched via a RAID array, which can either:

- ...stripe them together (i.e. effectively making 4x 2TB drives into one giant 8TB drive);
- ... OR mirror them, for an instant backup in case one drive in the formation fails. (Making 4x 2TB drives into two sets of 4TB drives, with one acting as a live backup in case the other set goes down).

Drives should also be purchased according to their type and the compatibility with the motherboard. Nearly every motherboard these days supports SATA, the new standard for drive connectivity; however there are multiple types of SATA: 1.5GB/s, 3.0GB/s and the newer 6.0GB/s. If your motherboard supports 6.0GB/s, and you plan on hosting/moving very large files with your server, it would be worth it to consider 6.0GB/s SATA drive(s).

Finally, brand name and warranty does still mean something, especially since hard drives are such important components in your server. After all, all your personal data rests on them; replacing the drive is much easier than replacing the data. Go with a brand that is known to be good. Western Digital Black series drives have a good record of dependability; many of them also come with record 5-year warranties, making them an excellent option.

## Other Stuff

Other things you will need to consider:

- Keyboard/Mouse
- CD/DVD drive
- Power strips and plugs
- Monitor: *Remember that this is optional if you are going to run a headless server, but you will at least need access to one temporarily when you install your distribution.*

### 3.2.3 - ISP and Domain Name Options



If you are not planning to use your server to host any external (Internet) service, OR you have opted to use a Virtual Private Server (VPS), you can skip this section.

Dealing with your internet service provider, no matter how much you might dread it, will be a necessary component of this setup if you plan on hosting a website or your email on this server. Your server needs the ability to be linked to a domain name, which means it also needs a static IP. This is something your internet service provider can give you. If you want to host multiple servers and services on VMs (say a fileserver VM and an email/web host VM) it would be a good idea to also get a static subnet.

Usually when you connect to the Internet, your service provider gives you a dynamically-set IP address to use. However when your web/email services go live, the Internet will need a steady and static address with which to look you up. This is why at least one static IP address is required. A static subnet is an extension of the above idea, but it obtains multiple static IP addresses that belong to a specific "subnet," or a subset of IP numbers. For example, if you were to obtain what is called a "/29 subnet," that gives you six static IP addresses to use.

Some residential internet providers no longer allow clients to request static IP addresses or subnets; if this is the case, you may need to consider springing for a Business class plan, as these always have the ability to obtain static IP addresses. In many cases they are not more than \$10 or \$15 more than your original residential plan would be.

Once you've dealt with your ISP, you must purchase a domain name. This will likely be much easier (and probably cheaper) than the prior step. There are many decent domain name

registrars out there, but I have to recommend NameCheap.com. As far as price, ease-of-use and customer service are concerned, they are consistently cited as one of the very best. For a domain, you can choose anything with any ending; though something simple is advisable if you are to be using an email address as well. Nothing like typing a 15-character domain when you want to send someone an email.

When buying a domain name, keep in mind that the domain you purchase will be subject to the laws and regulations of the country that you register it in. Wikipedia ran into trouble in the United States when its ".org" address was rescinded by US authorities because it published material that the government wasn't too happy to see. The common ".com," ".net" and ".org" are overseen by the US Government. Other countries, such as Iceland, have a more favourable policy towards the publishing of controversial or leaked information that would be in the public interest. It's advisable for those who look to post potentially sensitive information to consider an Icelandic domain. For more information regarding Iceland's national freedom of expression policy known as the "Icelandic Modern Media Initiative," [visit its website](#).

With the static IP in hand and the domain name registered, it's time to get them linked together. On your domain registrar's account page, there will be a place marked something like "Host Records" or "Domain Settings." (On NameCheap it is found at My Account > Manage Domains > click the domain name > All Host Records.) You will be presented with a list of fields, usually arranged into at least four columns: Host Name, IP Address, Record Type, and TTL.

- In the Host Name field "@", put your static IP address in the correct field, and set the record type as "A". This will allow people to reach your website by visiting `http://mydomain.com`.
- If there is a field for "www" hostname, or if you can create one yourself, do the same for an A record with your same IP address. This will allow people to reach the same site when going to `http://*www*.mydomain.com` as well.
- Finally, we will set our domain up for mail. There should be a section for "MX Records" or "Mail Settings." The hostname should be "mail", the IP address matching your static IP, and the "MX Pref" should be "10". When an email server wants to forward you an email, they will check this record and see your IP, allowing them to actually make the connection between servers and deliver the message.

With the correct settings enabled, and the Internet ready to welcome our server, you are ready to start assembling the server itself.

## 3.3. Assemble Your PC

---

This section will be included in guide version 1.5, due out in May 2013.

## 3.4. Installing Ubuntu Server

---

### 3.4.1 - Download Ubuntu Server

Downloading Ubuntu Server is a snap: you merely have to choose the version that is right for you. There are usually two different versions available at any given moment: the most up-to-date version (currently 12.10) or the current Long-Term Support (LTS) version, which is presently 12.04.1. It is usually a good idea to stick with the LTS version, as long as it is recent. This guarantees that you will be able to get support through Canonical (Ubuntu) for the foreseeable future, should you have a problem with the specific version you are using. Though this means you will not get the latest and greatest updates from Ubuntu, on server distributions this is usually not a problem.

Head to [Ubuntu Server's download page](#) and select the version that works best for you. Make sure to choose the correct architecture (32-bit or 64-bit) based on your server. Once you have the iso in-hand, burn it to disk with your preferred CD burning application.



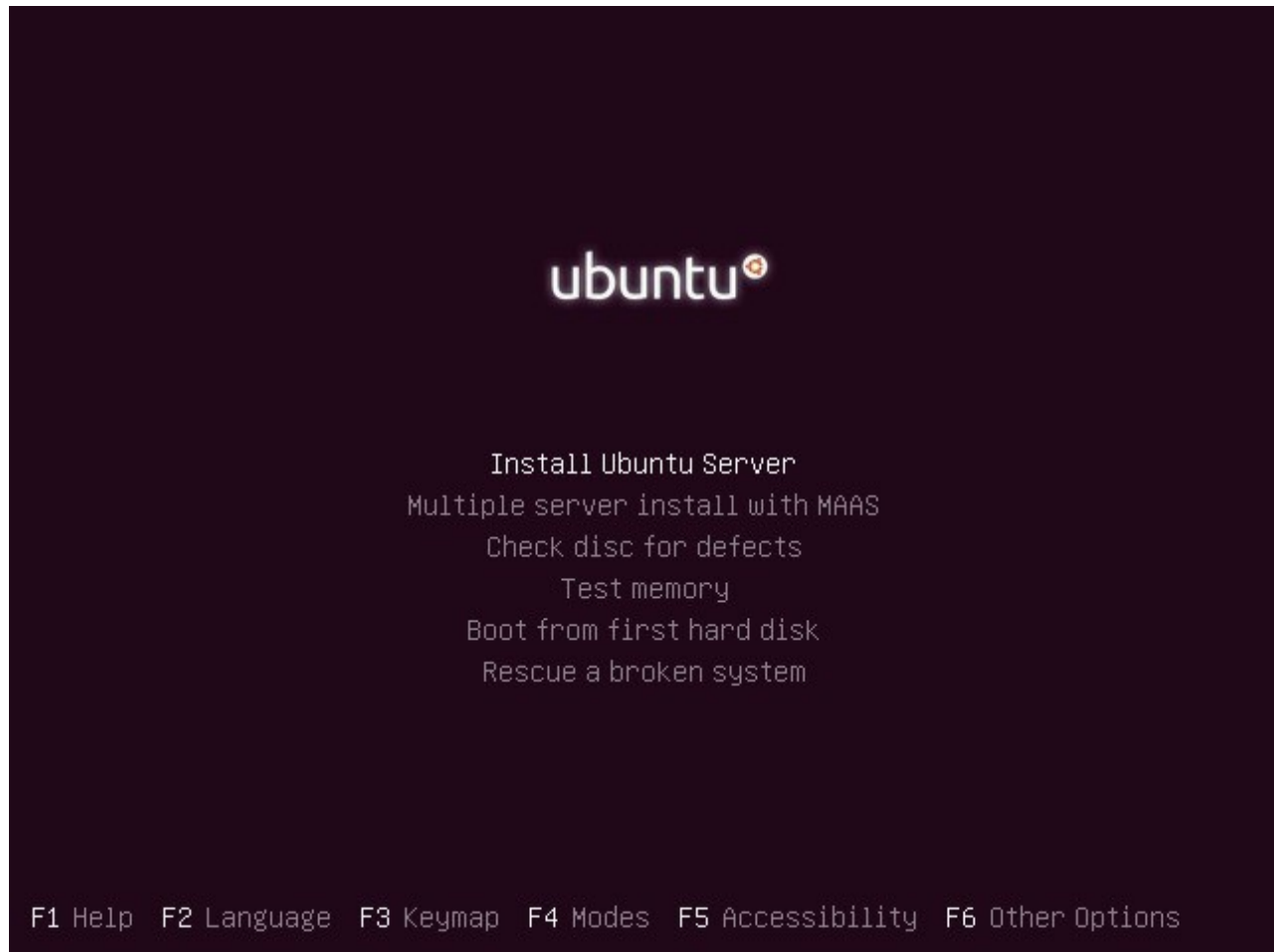
It is also possible to install Ubuntu via [USB drive](#).

### 3.4.2 - Installing Ubuntu Server

Installing Ubuntu Server is just as easy as installing Ubuntu's desktop version, but there are (of course) different options you will need to configure. Also, the installer is only available in a text-based menu format. You will be able to use the SPACE key to mark selected option buttons or checkboxes, and TAB to move between fields, just like in any other graphical application.

Load your install CD into your server, and boot it up, after having made sure that your CD/DVD drive is higher in the boot order list. Choose your language and the obvious options from the screen that comes up. It will ask you more questions based on your language, locale and other preferences.

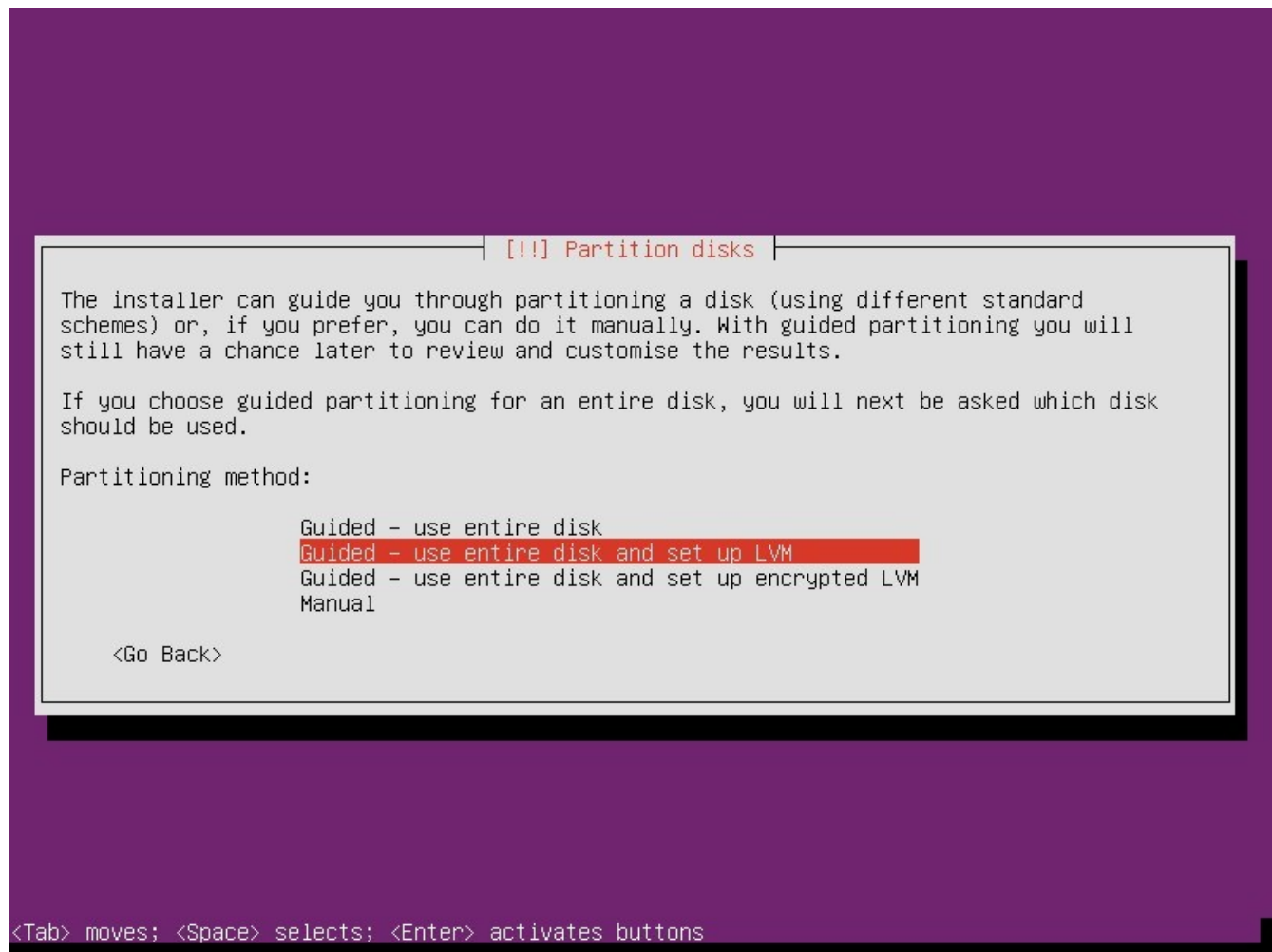




After this it will attempt to detect your hardware settings and will ask you if you wish to use DHCP. If your server is connected to a network that has a router, choose to use DHCP for now. If not, choose "Configure the network manually" and you will have the option to set your desired static IP, subnet and gateway settings. After the install, we will walk through specific network settings to enable based on your configuration.

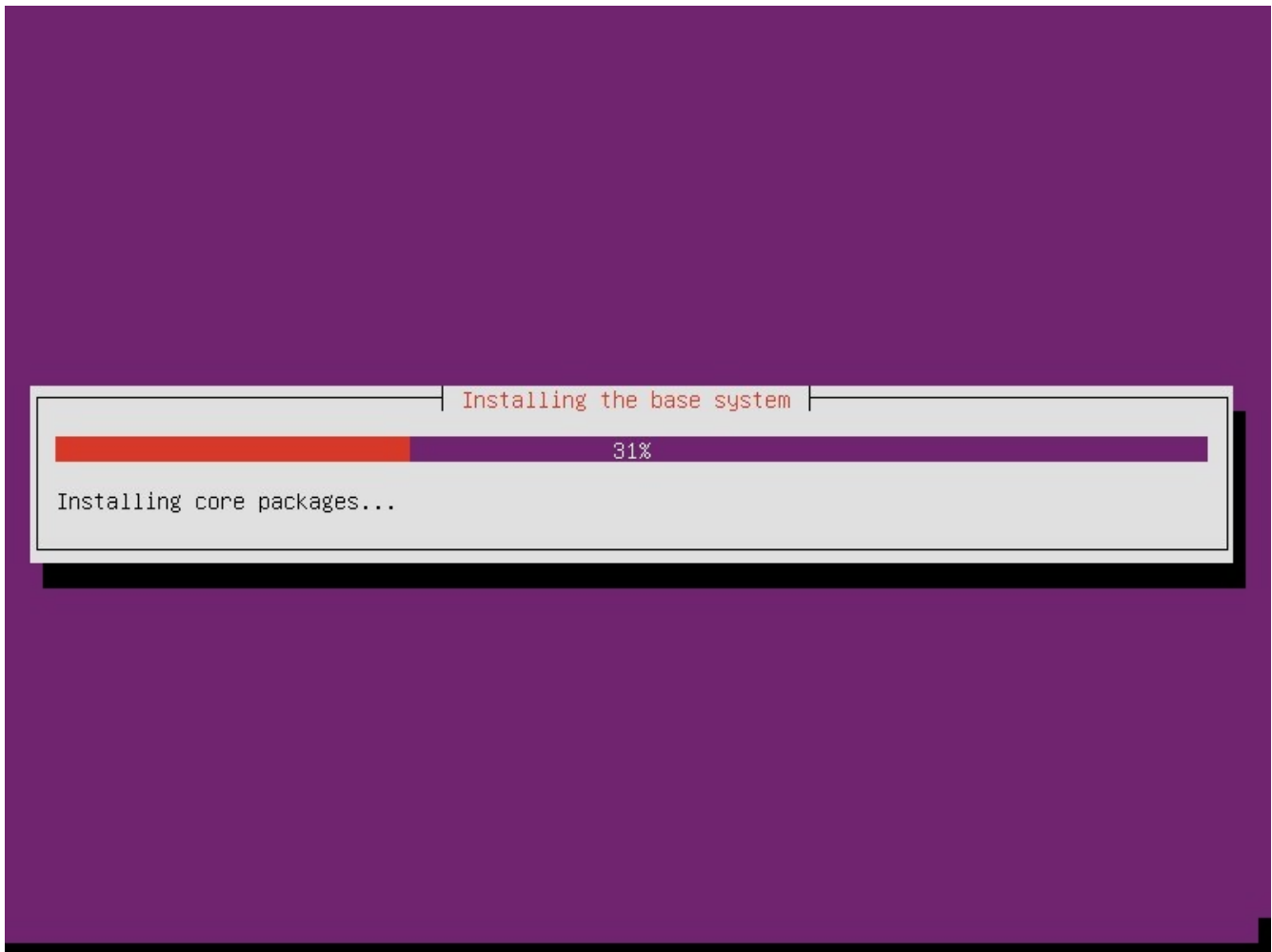
Set the hostname and timezone information as per your preferences.

Next the installer will take you to the disk configuration menu.



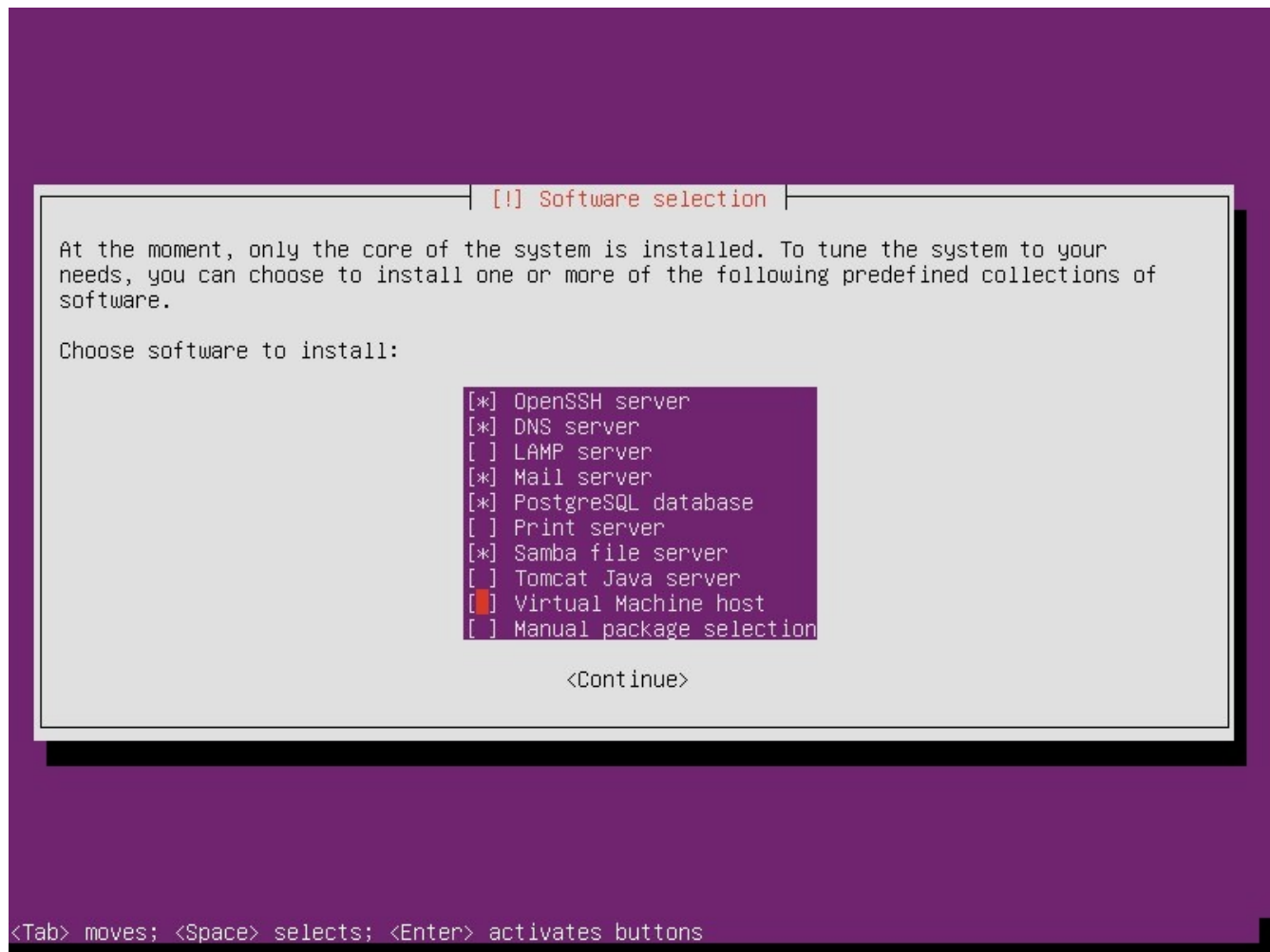
You will most likely want to choose "Guided - use entire disk". If this system will be running virtual machines or will share disk space with other operating systems, choose "Manual" and create a partition for "/" that reflects the size you want your server storage to have.

After this, your base system will be installed:



Next, you will set up a base user and choose its password, as well as setting the administrative password for the root user; then you will be asked if you want to encrypt the Home directory on the server. Unless you have extremely sensitive security concerns, I would not bother with encrypting the home directory on a server. We will be encrypting our backed-up data before we place it on the server anyway. Then you will be asked to choose how you want to receive your updates: either manually or automatically. Choose based on your preference. It is often convenient to have your server automatically receive security updates, so you don't need to worry about it.

Finally, you will be asked which software packages should be installed by default:



1. **OpenSSH Server:** It is highly recommended that you choose this. This will allow you to remotely access your computer from other machines, either on the local network or on the Internet. We will explain this in the next chapter, 3.5.
2. **DNS Server:** This is only necessary if you are going to use your server as a network controller and router. We go over this in chapter 3.6.
3. **LAMP Server:** This will install Apache (web server), MySQL (content publishing platforms like Wordpress or Drupal), and PHP (necessary for almost any website application). We will review these in chapter 3.9.
4. **Mail Server:** Installs Postfix and Dovecot for mail storage and transmission. We go over these in chapter 3.7.

5. **PostgreSQL Database:** This is another type of SQL server. You should only choose to install it if the program you want to run explicitly requires it.
6. **Print Server:** Use this if you will be connecting a printer to this computer and would like to share it on your network for other devices to use.
7. **Samba File Server:** Use this if you have Windows/Apple devices on your network that you will want to share files or media with. We will go over this in chapter 3.11.
8. **Tomcat Java Server:** This is for Java software hosting and development, you will not need it unless you are a Java developer.
9. **Virtual Machine Host:** Use this if you will be running virtual machines (VMs) with this server for various reasons. VMs will be explained in the appendix chapter 3.12.

And with that, your computer will reboot, and you will be presented with your shiny-new login prompt:

```
Ubuntu 12.04.1 LTS ubuntu tty1
ubuntu login:
```

This base system works according to the Linux command-line rules that were explained in section 2. It has no graphical user interface. The goal of this guide is to get you up-to-speed and comfortable with editing the features of your system without needing to rely on graphical interfaces.

### 3.4.3 - Basic Network Setup

At this point we will set up our server so that it has basic connectivity to the Internet. From there, we will be able to set up applications based on our individual preferences in the following chapters. Below we will explain how to set up your server to communicate with the Internet on one port, and with an internal network on the other. We will assume that "eth0" corresponds to the port connected to our internal network hub or access point, and "eth1" corresponds to the port directly connected to our DSL/satellite/cable modem.



If you have your server behind a router or other firewall which is handling your connection (and you will not be using the server itself as a router or firewall), you will need to assign the server a Static IP address on your router. This is necessary for various reasons. You will need to forward ports to your server for every service you will want to run from it, if you want to be able to reach them from the outside. Because of this, you will need to have the server on an internal static IP address that does not move, lest your running services be interrupted. In the steps below, you will also want to skip any settings for "eth1" as they do not apply.

First, you need to figure out the names of your network interfaces. Most of the time this will be "eth0" and/or "eth1," but to be sure, run `ip addr`. It will list the different interfaces you have. If you have two network interfaces, make sure you know which port corresponds to which by connecting them to different devices and monitoring how the `ip addr` entries change.



It is strongly recommended that you avoid running a server on a wireless interface (wlan0). For performance, stability and compatibility reasons, this is simply just a bad idea. This guide will not provide information on configuring servers connected wirelessly.

To set your server with a static IP address, open the file `/etc/network/interfaces` and add/change the following lines:

```
auto eth0
iface eth0 inet static
address 10.0.0.5
netmask 255.255.255.0
gateway 10.0.0.1
```

The "Gateway" should match the internal IP address of your internet-facing device (in most cases, your router). If this server is acting as a router/firewall and is directly connected to the internet with another ethernet port, set the gateway to be the same as the "address." The netmask will likely be "255.255.255.0", or a /24 subnet. Make sure the IP address you choose is on the same subnet as your existing network. That is, if your other devices all operate with IP addresses like 192.168.0.x, your server will need to be a static address in this range, HOWEVER it must be established outside of your router's DHCP address pool. Use your router's manual or online support to determine how to reserve a static IP address for a device.

Now we will add a section to the same file for our other ethernet interface, eth1. This port will be directly connected to our DSL/cable modem and will handle all internal/external requests for the Internet:

```
auto eth1
iface eth1 inet static
address 10.0.1.1
netmask 255.255.255.248
```

The "address" field will match the external static IP address provided by your internet service provider. The "netmask" must reflect the netmask of the static IP range you were given. If this is just one IP address, the netmask will be 255.255.255.252; if you received a Subnet from your ISP (like /24), you can convert that number to a netmask [with this calculator](#).

After setting these items, you will need to toggle the interface before the new settings take effect. Run `sudo ifdown eth0` then `sudo ifup eth0` to cycle the changes.



If your internet-facing ethernet port is connecting to a DSL modem, check to see if you connect to your DSL server via PPPoE. If this is the case, you will need to set up this ethernet port to connect to your modem via PPPoE. Follow the modem's manual or online support page to set it in "bridge" mode, then follow the [Ubuntu PPPoE guide](#) to set up the connection on your internet-facing ethernet port.

### **3.4.4 - Further Reading**

- [Ubuntu Installation Guide](#)
- [Ubuntu Server Guide - Network Configuration](#)



## 3.5. Getting In: Using SSH and VNC

---

Now that we have our server assembled and our OS installed, we must make sure we can get inside!

SSH is a protocol for secure communication between systems. It can be used for a wide variety of things, from executing commands on remote systems, to getting a remote terminal prompt on your local computer, to even running visual programs on a remote computer, but redirecting them to show up on your local computer's screen (X forwarding).

For the purposes of this guide, we will want to set up SSH and get comfortable with using the terminal remotely. If you are running a headless server, this is going to be your best friend.

### 3.5.1 - Install OpenSSH

Chances are that our Ubuntu Server came with OpenSSH already installed (that's how important it is!), but in the off-chance it hasn't, fire up your trusty-dusty package manager and install it:

```
sudo apt-get install openssh-server
```

Most of the configuration for OpenSSH is stored in `/etc/ssh/sshd_config`. This is your first stop for any additional configuration options, such as denying root login or allowing public-key authentication.

The great thing about SSH is that (in most cases) it works right out of the box. First, make sure the server is running:

```
sudo service ssh restart
```

Next, on your local computer, make sure you have a valid SSH client. (This is the package `openssh-client` on Ubuntu.) To test your setup, use the following command with the correct information:

```
ssh $username@$ip-address
```

After this you will get a prompt asking for your password. Once you enter it, you should get a command prompt as if you were using the terminal on your server locally. Voila! Type "exit" when you want to get back to your local computer's command prompt.

### 3.5.2 – Securing SSH

#### No Root Logins!

In its current state, your SSH is actually quite risky. Unless you laugh in the face of danger, you will want to take some steps to secure it.

First, we will prevent root SSH logins to our server. This is a popular line of attack – people (scripts) hoping to find just that *one* server that got lax and lazy with its configuration. We won't fall for that, obviously.

Edit your `/etc/ssh/sshd_config` file and change the following line:

```
PermitRootLogin      No
```

... then restart your SSH server. If you need to SSH into your server and change files/configurations that require root access, then you can SSH in as your normal user and use `su/sudo`, just like you would if you were working directly.

#### SSH Key Setup

What follows is completely optional but highly recommended. There is a way to set up a cryptographic key called an "SSH key" that will allow our computer to handle SSH connections without needing you to enter your password. There are two main reasons why people opt to use SSH keys:

1. **Security** - Even if you have what you might consider to be a "good" password, if somehow that password is guessed or compromised then there is a lot of potential risk. With an SSH key, you can actually turn off password logins, meaning that people

remotely won't even get a chance to *try* to crack your password. If they don't have your SSH key, then they're out in the cold.

2. **Laziness** – Like I said, SSH keys allow you to SSH to your remote machine without having to use your password. So if you are someone who needs to SSH to your server frequently, it can be a pain having to enter your password every so often. Much easier to let your SSH key do the talking for you – if your computer can produce the right key, the server will never ask you for a login password.

When you create an SSH key, you are creating two files: a **private** key and a **public** key. The private key is the actual file that is used to authenticate you. The public key contains a string that the server can use to compare with the private key and verify if it's really you trying to login. The private key is the one you do not want to lose.

To create an SSH key, run the following command on your **client** machine:

```
ssh-keygen -t rsa
```

This will ask you a few questions. First, go ahead and save it in the default location. Second, it's a good idea to enter a passphrase with which to unlock your SSH key. This is intended to provide a good last line of defence: should your SSH key somehow fall into the wrong hands, they still won't be able to get into your server. (Don't worry, if you set a passphrase here, you can still set it to automatically unlock itself on your own computer via `ssh-agent`.)

After you've created your key and given it a passphrase, run the following command with the correct information in place to upload it to your server:

```
ssh-copy-id $username@$servername
```

This copies your public key to an “authorized keys” list, telling your server that whichever computer SSHes in with your private key in hand can be trusted. The neat thing about this is that you can put your SSH private key on any computer you own (even your Android smartphone) and be able to gain password-less access to your server.

When you test your SSH connection, your client will automatically use your SSH key. It should only ask you for your passphrase the first time; if not, run the command ``ssh-add`` and it should be permanently added to your ``ssh-agent``.

It should go without saying that it's very important this key be kept secure. I would recommend storing a backup on a USB key that you can hide somewhere in your home with your personal files. And if you store it anywhere else on your computer/server, like in a backups folder, make sure you store it in an encrypted archive (see the chapter on Backups for how to do that).

## Use Your SSH Key On Other Devices

If you wish to use your SSH key on...

- ...Other Linux machines **OR** Mac OS X: Copy `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub` to the same folder on your other Linux computer. Run ``ssh-add,`` then voila.
- ...A Windows computer (hisssssss): Download [PuTTY](#). Enter your hostname/IP in the first section, then choose "SSH." In the menu on the left, choose SSH > Auth. Browse to the location of your private key, click OK and start the session.
- ...an Android phone: Copy your `id_rsa` file to your phone (in a preferably secure location) via your file transfer method of choice. Download [ConnectBot](#) from the Play Store and install it. Open the app, press Menu and choose "Manage Public Keys." Press Menu and choose "Import," then browse to the location of the file and choose it. Note that when you create a new connection, you can hold down the line in the list and choose Edit Server, then explicitly set that you wish to use the key for that connection. This provides the best results.

### 3.5.3 - Install VNC

VNC is another way to remotely gain access to your computer. Where SSH gets you into the terminal, VNC is a more direct approach. It resembles the "Remote Desktop" application on Windows systems.

This protocol is only worthwhile for servers with graphical interfaces, like the full version of Ubuntu. If you are using the Ubuntu Server we have been talking about, you will be better off sticking to SSH.

Ubuntu comes with a built-in VNC server called `vino`. It is enabled by default.

On your local machine you will need a VNC viewer. Ubuntu has one built-in named `vinagre` that will work nicely for our purposes. From the command line, enter the following with your server's IP address:

```
vinagre 192.168.0.1
```

When it comes to securing your VNC connection, the best way to do that is to run VNC over an SSH tunnel and block the VNC port (5900) on your firewall. We will discuss port blocking and SSH tunnelling in chapter 3.10.

### **3.5.4 - Further Reading**

- [OpenSSH Server - Ubuntu Server \(12.10\) Official Documentation](#)
- [ssh\\_config man page](#)
- [VNC - Community Ubuntu Documentation](#)

## 3.6. Home Networking: DHCP, DNS and NAT

---

For those who will be using their servers to manage their network (including as a firewall), we will now be setting up various services allowing our internal network to use the Internet and various other services hosted by our server.

### 3.6.1 - Serve Network Clients via DHCP

First, install the DHCP server from the Ubuntu package repositories.

```
sudo apt-get install isc-dhcp-server
```

Now, to configure it, we will create several customized entries in `/etc/dhcp/dhcpd.conf` to handle our setup.

```
default-lease-time 432000;
max-lease-time 604800;
option routers 192.168.0.1;
option domain-name-servers 192.168.0.1;
option broadcast-address 192.168.0.255;
option subnet-mask 255.255.255.0;
option domain-name "$home.local";

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.10 192.168.0.50;
    host $myhost {
        hardware ethernet xx:xx:xx:xx:xx:xx;
        fixed-address 192.168.0.x;
        option host-name "$Myhost";
    }
}
```

Now let's walk through these lines and figure out what each of them does.

- **default-lease-time** and **max-lease-time** govern how often your computers will check back with the server to have their IP address assignment renewed. The figure is in seconds. In the majority of cases, you can set this to be a somewhat long time and there will be no issues. If you set the leases to be too short, it may impact your network performance. 432,000 seconds equals 5 days.
- **option routers** and **option domain-name-servers** needs to point to your server's static IP address, that you gave it in the Server Installation chapter.
- **option broadcast-address** is for the internal network broadcast address. The last octet (set of numbers) should always be 255. If your network is in the 192.168.1.x range, then change the 1. Otherwise it should be left alone.
- **option subnet-mask** should be left at its default, 255.255.255.0. If you need a different one, it's likely because you have a huge network with hundreds of computers; if that's the case, then you shouldn't be following this guide anyway :)
- **option domain-name** should match what you chose as your internal domain name. In most cases, "home.local" will suffice.
- **subnet 192.168.0.0 netmask 255.255.255.0 {** begins the section that outlines the internal network we are now setting up. The first IP address (192.168.0.0) combined with the second number (255.255.255.0) means that all of our clients will have IP addresses that begin with 192.168.0, AND that we can add any number at the end of that from 0-254 for network clients.
- **range 192.168.0.10 192.168.0.50** is important, because it tells the DHCP client how many addresses in the 192.168.0.0 block it can claim as its own and assign to clients. Its usually a good idea to have a bit more than you need here; as you are not likely to have over 200 machines on this network, than you won't be needing to worry about space.
- The next nested section (**host \$myhost**) is optional. If you want one of your computers to always receive the same IP address via DHCP, which is convenient for diagnostic purposes and is recommended for any other servers running on your network. Replace the hostnames listed here with what they should be for that computer. Set the MAC address to the network adapter that the computer will connect from. (On Linux-based systems you can usually find the MAC address by running `ip addr`.)
- And finally, don't forget to close out all the open sections you opened with "{" with a corresponding "}"!

Once your configuration is in order, start the server with `sudo service isc-dhcp-server restart`. Your devices will now be able to communicate with each other on your network. But don't get too excited yet! They still won't be able to get internet access. For this, we will need to set up a gateway and NAT forwarding with iptables, then we will set our server to handle DNS requests.

### 3.6.2 - Give Clients Internet Access with iptables

The next step is to enable your server as an Internet gateway, so that it will share its connection to devices connected to the internal network. To do this, we will be using the iptables firewall system.

```
sudo iptables -A FORWARD -o eth0 -i eth1 -s 192.168.0.0/24 -m
conntrack --ctstate NEW -j ACCEPT
sudo iptables -A FORWARD -m conntrack --ctstate
ESTABLISHED,RELATED -j ACCEPT
sudo iptables -t nat -F POSTROUTING
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables-save | sudo tee /etc/iptables.sav
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```



"-o eth0" should match your outside interface (connected to your modem), and "-i eth1" should match your inside interface, connected to your hub or access point.

Set your iptables configuration to load at boot by editing `/etc/rc.local` and adding the following line:

```
iptables-restore < /etc/iptables.sav
```

Finally, edit `/etc/sysctl.conf` and uncomment the line that reads `net.ipv4.ip_forward=1`.

And with that, our iptables configuration should be working. We will work more with iptables in the chapter on firewalling and security, 3.10. At this point your devices should now be able to ping IP addresses that are on the internet, and view internet sites via IP addresses. But the final piece of the puzzle comes in handling DNS requests.



### 3.6.3 - Set Up a Local DNS Server

In brief, DNS is the method that the Internet uses to translate IP addresses to the domain names we are all used to typing in our browsers. We know that every internet server has at least one IP address, and this is how it can be "found" online. And DNS is what is used to give these addresses a human-readable name.

Our server will be set up for DNS for two purposes:

- **Caching:** For every page request made to the Internet from one of your computers, the server will keep a cache of its location data. You may notice that the first time you view a site, it is often slower to load than the subsequent times you visit it. This is subsequently due to your computer "seeking" the address of the server the first time; every time after that, it will remember where it went before. Setting your server to act as a DNS cache locally should improve internal network performance overall.
- **Internal Authority:** This DNS server will keep track of the device names on our network, and allow other devices to be able to find them by those names. So if you want to SSH to your computer in the other room, you can do so by running `ssh ComputerName` instead of having to keep track of its IP address at any given time and running `ssh 192.168.0.?`.

The DNS server we will use is called BIND. Install it by running `sudo apt-get install bind9`.

To configure BIND as a caching nameserver, edit `/var/lib/bind/named.conf.options` and change the following lines:

```
forwarders {
    x.x.x.x;
    x.x.x.x;
};
```

The x.x.x.x lines should match the Primary and Secondary DNS addresses given to you from your Internet Service Provider. If you do not have any or do not know what they are, you can use 8.8.8.8, which forwards to Google's public DNS servers.

Now we will set up our DNS server to act as our internal network's authority. This comes via setting up two zonefiles. Create a file named `/var/lib/bind/db.home.local`. (Change the trailing "home.local" to whatever you decided your internal domain would be earlier.)

Enter the following in this file, replacing the \$ values where appropriate EXCEPT leave the \$TTL and \$ORIGIN as they are:

```
$ORIGIN .
$TTL 86400
$home.local      IN SOA      $myserver.home.local.
$username.home.local. (
                        2012112301      ; serial
                        28800            ; refresh (8 hours)
                        14400            ; retry (4 hours)
                        2419200          ; expire (4 weeks)
                        86400            ; minimum (1 day)
                    )
                    NS      $myserver.home.local.
                    MX      10 $myserver.home.local.
$ORIGIN home.local.
myserver          A      192.168.0.1
laptop            A      192.168.0.2
workstation       A      192.168.0.3
phone            A      192.168.0.4
xbox              A      192.168.0.5
```

The third line (starting with "home.local") should feature your internal domain. The next bit (myserver.home.local.) should reflect your server's hostname with the internal domain and a "." appended to the end. The last bit on this line (username.home.local.) is actually an administrative email address - change this to match the email you want to use for this field, making sure there is a "." in the place of the "@", and a "." at the end of it all.

The NS and MX lines should point to your server's hostname and internal domain. This is used to designate the server as the internal domain's nameserver and main mail server.

The repeated entries below the second \$ORIGIN tag are individual records for devices on the network. These are called "host entries." Remember when, in our DHCP configuration, we had the opportunity to reserve specific addresses based on the MAC addresses of our devices? These same entries should be repeated here, with the accompanying "A" tag in the middle. Now we don't need to add entries for every possible device we will have on our

network here: in the next section we will have DHCP do this for us. But it is a good idea to include your server in this list, as well as anything you've given static or reserved IP addresses.



Whenever you change a zonefile, you **must** increase its serial number. Many people use the date in YYYYMMDD format, then a couple digits marking the number of the change you've made.



There are many other kinds of host entries you can make here; for information on them see the BIND links in the Further Reading section.

Now for every DNS zonefile we establish, we must have a corresponding "reverse DNS zonefile." This is fairly simple to do; create a file called `/var/lib/bind/db.192` and insert the following, replacing the \$ values where appropriate EXCEPT leave the \$TTL and \$ORIGIN as they are:

```
$ORIGIN .
$TTL 86400
0.168.192.in-addr.arpa  IN SOA      $myserver.home.local.
$username.home.local.  (
                        2012112301      ; serial
                        28800           ; refresh (8 hours)
                        14400           ; retry (4 hours)
                        2419200         ; expire (4 weeks)
                        86400           ; minimum (1 day)
                        )
                        NS      $myserver.home.local.
$ORIGIN 0.168.192.in-addr.arpa.
1          PTR    myserver.home.local.
2          PTR    laptop.home.local.
3          PTR    workstation.home.local.
4          PTR    phone.home.local.
5          PTR    xbox.home.local.
```



The "0" in "0.168.192.in-addr.arpa" refers to the third octet in your network's IP subnet. It assumes your network operates on the 192.168.0.0 range. If it is otherwise, update this number accordingly.

Now a lot of these options are customized in the same way they are in the first zonefile we made, but we can see a pretty important difference when we get down to the host records. They are in reverse order. The last octet of the IP address for each device (e.g. the "1" in "192.168.0.1") is placed first, followed by the "PTR" (pointer) flag, then the fully-qualified hostname with internal domain appended at the end. Remember that you only need to create records here if you created them in your first zonefile, and you don't need to create records for *every* device on your network.

To activate these zonefiles for use in BIND, edit `/etc/bind/named.conf.local` and add the following lines:

```
zone "home.local" IN {
    type master;
    file "/var/lib/bind/db.home.local";
};
zone "0.168.192.in-addr.arpa" {
    type master;
    file "/var/lib/bind/db.192";
};
```

*Whew*, are you still with me? DNS setups can be a real headache, but if you've made it this far with your sanity intact, then you are almost ready to reap the rewards!

Start up bind with `sudo service bind9 restart`. At this point, your clients should be able to connect to the Internet using regular ol' domain names like usual. Hooray!

### 3.6.4 - Allow DHCP to Update DNS Entries

Now we can not only use the Internet on our internal network, we can also communicate with our static servers/hosts using their proper names. But what if you want to reach other devices by their hostnames? Say you have a friend come over that's bringing his laptop, and you want to set up a fileshare on it and to reach that share via his laptop's hostname. For that, we can allow our DHCP server to fetch these names and update our network's DNS records accordingly. This is done by providing a secure socket for the DNS and DHCP servers to communicate on.

First, change the owner of your zonefiles to let BIND be able to edit them at will:

```
sudo chown bind:bind /var/lib/bind/*
```

Now we will generate a key that will allow the two programs to communicate securely between each other.

```
sudo cat Kdhcp_updater.*.private | grep Key
```

Copy the output or write it down; we will need it soon. Open up `/etc/bind/named.conf.local` again and add the following lines:

```
key DHCP_UPDATER {
    algorithm HMAC-MD5.SIG-ALG.REG.INT;

    Important: Replace this key with your generated key.
    Also note that the key should be surrounded by quotes.
    secret "asdasddsaasd/dsa==";
};
```

While in `named.conf.local`, add the following line inside the brackets for each zone you have declared there:

```
allow-update { key DHCP_UPDATER; };
```

So we are set up on the DNS end, now let's give DHCP the other end. Edit `/etc/dhcp/dhcpd.conf` and add the following to the very top of the file:

```
ddns-domainname "$home.local.";
ddns-rev-domainname "in-addr.arpa.";
ddns-update-style interim;
ignore client-updates;
```

Next, add the following before the "subnet" section:

```
key DHCP_UPDATER {
    algorithm HMAC-MD5.SIG-ALG.REG.INT;

    Important: Replace this key with your generated key.
    Also note that the key should be surrounded by quotes.
    secret "asdasddsaasd/dsa==";
};

zone home.local. {
    primary 127.0.0.1;
    key DHCP_UPDATER;
}

zone 0.168.192.in-addr.arpa. {
    primary 127.0.0.1;
    key DHCP_UPDATER;
}
```

Accordingly, we will allow the DHCP server to write to its files:

```
sudo chown dhcpd:dhcpd /etc/dhcp/dhcpd.conf
```

Restart the servers with `sudo service bind9 restart` and `sudo service isc-dhcp-server restart`, and it's done!



Don't forget to remove the key file that we created, `Kdhcp_updater.*`.



From now on, if you want to make manual changes to your BIND DNS zonefiles, you will need to "freeze" them first. Freeze it with `sudo rndc freeze home.local.` and then you are free to make your edits. Once completed, "thaw" the zonefile again by running `sudo rndc unfreeze home.local.` And of course, don't forget the `."` at the end!

### 3.6.5 - Further Reading

- [DHCP \(Ubuntu Documentation\)](#)
- [BIND Configuration \(Ubuntu Documentation\)](#)
- [Internet Connection Sharing \(Ubuntu Documentation\)](#)
- [DNS Record Updates via DHCP - Lani's Weblog](#)

## 3.7. Host Your Email: Setting Up Postfix and Dovecot

---

There are two components to the mail system we are going to build. The first component is **postfix**. Postfix is what we call a "Mail Transfer Agent" (MTA). An MTA is responsible for transporting email between different destinations. When you open your email application and send an email, that document gets transferred first to your email provider's MTA. The MTA then parses the message for a destination address, looks up its server's location on the Internet, then facilitates the transfer of the message to that server. An MTA also handles incoming email in the same way: your MTA gets contacted with a message from somebody else, then your MTA delivers the message to the Mail Delivery Agent.

The Mail Delivery Agent (MDA) is the second part of the mail system. Our MDA is called **dovecot**. The MDA handles the storage and organization of your mail once it is received. It may help to think of it as such: your MTA is your postman, going from house to house and delivering the mail; the MDA is your mailbox itself.

### 3.7.1 - First Steps: Install Postfix

We will begin with installing our MTA, Postfix.

```
sudo apt-get install postfix
```

Postfix comes with a handy semi-graphical configuration tool, which we will use to start. Run the following:

```
sudo dpkg-reconfigure postfix
```

Fill in the following details, which will match our configuration.

1. **Mail server configuration type:** "Internet Site".
2. **System mail name:** *mydomain.com*
3. **Root and postmaster mail recipient:** *leave blank*
4. **Other destinations to accept mail for:** Add *mydomain.com* to the beginning of this comma-separated list.
5. **Force synchronous updates?:** No



6. **Local networks:** Enter your IP subnet that we picked in the Networking section.
7. **Use procmail?:** No
8. **Mailbox size limit:** "0"
9. **Local address extension character:** Leave as default.
10. **Internet protocols to use:** all

Now we need a place to put all that mail that's sure to arrive. In this example we will use the Maildir format, so run the following with your username in the place of \$username:

```
sudo postconf -e 'home_mailbox = Maildir/'  
export MAIL=/home/$username/Maildir  
sudo postfix restart
```

And with that, we have a simple mail transport system running! Take a moment to pat yourself on the back.

Now we will test what we have just set up. Ensure that postfix is running with `sudo postfix status`. If it's not, run `sudo postfix start`. Then open up `telnet` and open a session to your local SMTP port:

```
telnet localhost 25
```

You'll receive the following output and a prompt if you have successfully connected:

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

This prompt is a little different from the standard command, as it only understands SMTP commands. But not to worry - enter the following commands line-by-line to send yourself a test message:

```
ehlo localhost
mail from: root@localhost
rcpt to: $username@localhost
data
Subject: My Postfix Test

Test Message 123
This is the body
Goodbye
.
quit
```

Make sure to put your username in the right spot. Also, that line right above "quit" is indeed just a period. That tells postfix that our test message is complete and ready to be sent.

Now let's see if it worked. Run the `mail` command and you should see the subject line of your message. Press 1 and Enter to read it. Postfix is aliiiiiiiiiiiiive!



In most cases, mail clients will send their outgoing mail on port 25. This is the port that mail servers communicate between each other with to transfer mail. However, many mail clients are set up by default to use port 587 to send mail over Secure (TLS) SMTP. If you'd like, you can also enable this port in Dovecot by editing `/etc/postfix/master.cf` and uncommenting the line that starts with "submission." To require logins over TLS for this port, uncomment the `smtpd_recipient_restrictions` section underneath "submission" and add `reject_sender_login_mismatch` to the list.

### 3.7.2 - Setting Up Mail Storage with Dovecot



This guide assumes that you want to run a mailserver for personal use only. It will therefore base your mail account off of your login account. If you are planning on running a server for others as well, it would be a good idea to set up virtual users, instead of setting up multiple users on your computer itself and potentially compromising it. To enable virtual users, follow the steps outlined in this guide, then add the steps found [here](#).

On Ubuntu Server, there are two flavours of dovecot: `dovecot-imapd` and `dovecot-pop3d`. You can install either or both if you'd like. Though the one you choose will depend on which email protocol you would like to use for remote connections. POP is the older protocol, which operates by downloading all email on a remote server to local folders and organizing them by their type. POP then deletes the original messages from your server, leaving you with the copies and folder organization on your local computer only.

IMAP, on the other hand, is a more robust system and is recommended for those who prefer to have their mail synced to multiple locations (for example, on your laptop and on your mobile phone). IMAP syncs your mailbox's folders between the client and the server. Whenever you move an email between boxes, for example, IMAP will sync those changes to your email server in real time. You should be able to see how this is beneficial to people who use their email on multiple devices: no matter what you read or where you read it, the email's status and location can be synced across all of your devices.

So choose the version(s) of dovecot you would like to install:

```
sudo apt-get install dovecot-imapd
```

Your main dovecot configuration file is stored at `/etc/dovecot/dovecot.conf`. In some versions of the software, including newer Ubuntu versions, this file "includes" other configuration files stored elsewhere, which can be found in `/etc/dovecot/conf.d`. We will be editing a variety of files to get our mail storage system set up.

Let's start with setting up our Maildir. This is the spot where mail is temporarily stored as dovecot routes it to its proper destination. Change the `mail_directory` line in `/etc/dovecot/conf.d/10-mail.conf` (or `/etc/dovecot/dovecot.conf` to match:

```
mail_location = maildir:/home/%u/Maildir
```

Now we will set up the mail storage hierarchy and enable it for use with the following commands, again changing `$username` for the appropriate value:

```
sudo maildirmake.dovecot /etc/skel/Maildir
sudo maildirmake.dovecot /etc/skel/Maildir/.Drafts
sudo maildirmake.dovecot /etc/skel/Maildir/.Sent
sudo maildirmake.dovecot /etc/skel/Maildir/.Trash
sudo maildirmake.dovecot /etc/skel/Maildir/.Templates
sudo cp -r /etc/skel/Maildir /home/$username
sudo chown -R $username /home/$username/Maildir
sudo chmod -R 700 /home/$username/Maildir
```

Once this is complete, we are ready to start and test Dovecot. Start it with `sudo service dovecot start`. Then open up a telnet with `telnet localhost imap`. If you see something like this:

```
Trying localhost...
Connected to localhost.
Escape character is '^]'.
+OK dovecot ready.
```

... then we are ready to go to the next step!

### 3.7.3 - Securing Your Mail System

The importance of running a safe and secure mail system cannot be overstated. For one, you certainly don't want your system to be used to forward spam off across the internet. If your system allows spam to be relayed then it can even find its way onto a blacklist, meaning some providers can refuse mail from your email accounts! And we certainly don't want that. So it is very important that we secure our mail system. To do this, we will enact the following policies:

First comes our Postfix SASL configuration. This is the mechanism that Postfix uses to securely authenticate users and servers. You will need to install the `libsasl2-2`, `sasl2-bin` and `libsasl2-modules` packages. Now, open up `/etc/default/saslauthd` and change the following lines:

- `START=yes` should be uncommented
- Add or change the following lines:
  - `PWDIR="/var/spool/postfix/var/run/saslauthd"`
  - `PARAMS="-m ${PWDIR}"`
  - `PIDFILE="${PWDIR}/saslauthd.pid"`
  - `OPTIONS="-c -m /var/spool/postfix/var/run/saslauthd"`

Run the following commands to enable SASL in your postfix configuration:

```
sudo postconf -e 'smtpd_sasl_local_domain ='
sudo postconf -e 'smtpd_sasl_auth_enable = yes'
sudo postconf -e 'smtpd_sasl_security_options = noanonymous'
sudo postconf -e 'broken_sasl_auth_clients = yes'
```

Next, we will set the access restrictions for sending mail on our server:

```
sudo postconf -e 'smtpd_recipient_restrictions =
permit_sasl_authenticated,permit_mynetworks,reject_unauth_destination'

sudo postconf -e 'inet_interfaces = all'
```

This line tells Postfix that our server will automatically accept mail from authenticated users (like your mail client), OR on any device connected to our own network, because we know they can be trusted. Furthermore, our server will outright reject any mail sent to it that is not addressed to our domain OR that is sent from our domain.

Finally, we will start up our SASL authenticator by running:

```
dpkg-statoverride --force --update --add root sasl 755
/var/spool/postfix/var/run/saslauthd
sudo service saslauthd start
```

At this point, were you to run `telnet localhost 25` and pass `ehlo localhost`, you should receive `250-STARTTLS` as one of the responses. That means secure logins are now available for our outgoing mail server.

Next we will set up our mail storage system (Dovecot) to allow clients to connect to it in a secure way. This will enable us to use encrypted connections when we are away from home, so no snoops will be able to pick out our passwords when we check our mail.

Edit the `/etc/dovecot/conf.d/10-ssl.conf` file, and change the following lines:

```
ssl = required
ssl_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
ssl_key_file = /etc/ssl/private/ssl-cert-snakeoil.key
```



If you are planning on running an email system for multiple people, it may be a good idea to use a purchased SSL certificate instead of a self-signed certificate. If not, all of your clients will get "Untrusted" messages before using their email, which may be unsettling. If you purchase these certificates, change the above pointers to match the location of the appropriate certificate and keyfile on your system. For more information on SSL certificates, check out the [LDP page on the subject](#), or see chapter 3.8 to hear them explained in the context of web servers.

With this, Dovecot will require its clients to authenticate themselves securely. You can now test your system by opening up your mail client of choice and adding your mail account. Enter the username and password of your user account on the server that you want to use. Set `mail.mydomain.com` as both your incoming (IMAP) and outgoing (SMTP) mail server. Make sure IMAP is using port 143, and SMTP is using port 25 or 587, whichever you chose in the Postfix configuration.

### 3.7.4 - Further Reading

- [Postfix Documentation](#)
- [Dovecot Wiki](#)

## 3.8. Host a Website with Apache and PHP

---

**Apache** is a free web server daemon that will enable you to host a wide array of websites, from simple landing pages for your contact information and resumé, to large e-commerce sites or content platforms. Together with **MySQL** database management, **PHP** scripting, and - of course - **Linux**, the "**LAMP**" stack is a popular starting base for running a wide variety of web applications and platforms.

We will start our web hosts' configuration with Apache. Apache's versatility is one of its best assets. It supports a range of modules that can be added on to expand its usability for different services or applications. New sites can be set up very easily, with the quick creation of VirtualHost file you can be up and running in seconds. As a result this may be one of the shortest guides on the site!

### 3.8.1 - Installing apache2

On our Ubuntu server, Apache was most likely installed by default when we chose to install our LAMP server. If for some reason you have no files under `/etc/apache2`, you can install Apache by running:

```
sudo apt-get install apache2
```

Once Apache is installed, make sure it is running with `sudo service apache2 restart`. You will be greeted with a lovely message like "It works!" if you navigate your web browser to the IP address of your server.

At this point you have a very basic web server. If you know HTML, you can create pages and place them in your default web directory, `/var/www`, and they will show up when you navigate to your domain name or IP address.

To create separate sites on different subdomains or for different services, you can create VirtualHost files to manage them. It will also allow you to activate or deactivate these sites in a modular way if you need to do some debugging. VirtualHost files are stored in `/etc/apache2/sites-available`. They can be enabled with `sudo a2ensite $sitename` and disabled with `sudo a2dissite $sitename`.

This is a sample format for a VirtualHost file:

```
<VirtualHost *:80>
    DocumentRoot /www/example1
    ServerName example.com
    ServerAlias www.example.com

    # Other directives here

</VirtualHost>
```

**DocumentRoot** is the physical location on your server that has the HTML/PHP/whatever files to be served. **ServerName** is one of your domains, but it can also have a subdomain attached. For example, if you wanted to host a site only to be seen at `http://secretsite.mydomain.com`, you could put "secretsite.mydomain.com" as your **ServerName**. **ServerAlias** makes a site available on more than one domain or subdomain. There are plenty of other parameters for VirtualHost files that you can use. See an intro to some of them at [Apache's documentation site](#).

While we can now set up websites via Apache with no problem, it's most likely that you will want to use another platform to manage your content, such as Wordpress or Drupal. These will allow you to automatically add blogs, photo galleries and other content to your site via a clean interface and no coding required. For those, we will need to assemble the next components of our LAMP stack.



### 3.8.2 - Adding Databases and PHP

## PHP

PHP is easy to install in Apache. To do so, run:

```
sudo a2enmod php
```

then restart Apache:

```
sudo service apache2 restart
```

With that, your Apache server will be able to parse and serve PHP files as normal.

## MySQL

Next, we can get to setting up our databases. There are many different database systems out there, this guide can't possibly cover them all. However, MySQL is the database system that is most frequently used for popular web applications and platforms. Both Wordpress and Drupal use MySQL. MySQL should have been installed with our Ubuntu Server, but if not you can install it with:

```
sudo apt-get install mysql-server
```

During the installation, you will be given the opportunity to set a root password for the MySQL user. Set this to something secure but accessible, as we will need it later to configure our database.

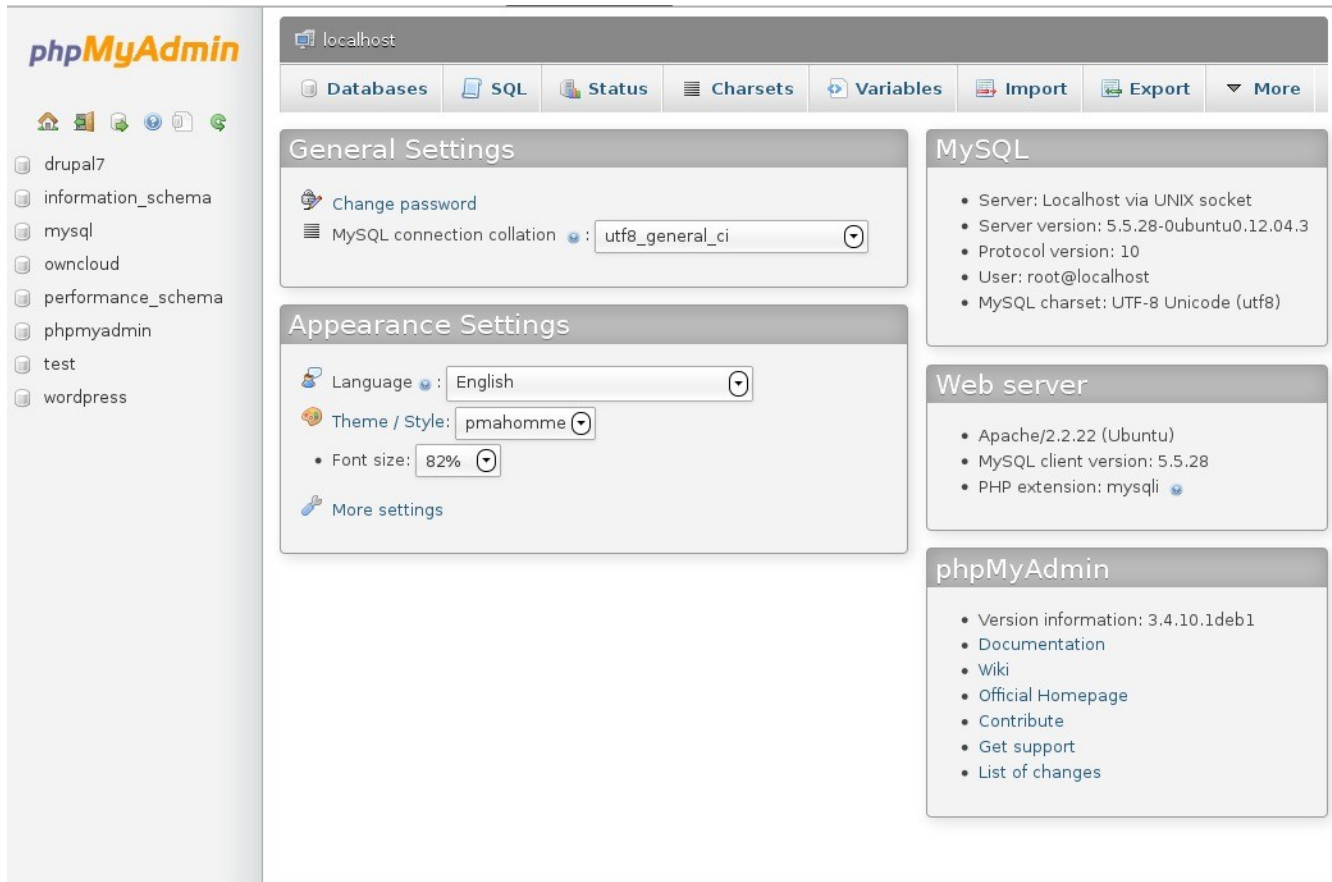
## phpMyAdmin

We will now install phpMyAdmin, which is a visual front-end to MySQL and will allow us to easily set up databases for our web apps. Run the following:

```
sudo apt-get install phpmyadmin
```

It will ask you what server to use. Choose Apache, as that is what we are using as a web server.

Next, head to `http://$your-ip-address/phpmyadmin` and we will continue our configuration from there. Log in using "root" as the username, and the root password we chose earlier. You will be greeted with a similar landing page:



From here you will be able to add new databases and users as necessary. In brief:

- To add a new database, click "Databases", and near the top of the screen you will see a field to enter a name and a Create button for your new database.
- To add a new user, which is recommended for most apps instead of giving them root database access, click "Privileges," then "Add a new User." You will be able to assign this user a specific database. Then you can use this user's name and password when your application needs database access.

Now that we know the basic ins and outs of our MySQL setup, we are ready to install a web application for our new server. You can choose any platform you like based on your needs and what you actually want to do with your server. As an example case, we will go through the installation of WordPress, a simple and easy-to-use blogging platform.

### **3.8.3 - (Optional) Install and Run WordPress**

WordPress, the wildly popular and effortless blogging platform, is available for installation in the Ubuntu repositories. However, the versions that are usually carried in distribution repositories are often out-of-date by at least a few versions. In order to have the most secure and cutting-edge experience, we will [download the source from WordPress directly](#), then install it to our webserver.

Unzip the install package to a folder of your choosing under `/var/www`. If you want the blog to be at the base of your webserver, such as `http://mydomain.com` with no subdomains or subfolders required, it is OK to unzip the package to the base `/var/www` directory.

After unzipping, you will need to set up a MySQL database and user that your WordPress installation can use. Go to `http://mydomain.com/phpmyadmin`, login with your root credentials, and set up a database using the instructions found in 3.8.2. The database can be named anything but usually just "wordpress" suffices. After that, set up a user (named anything, but "wp" seems to be a default). The user should have access to the new "wordpress" database.

Now we are ready for WordPress' "Famous Five-Minute Install." It might even take less time than that! Open up your web browser and go to `http://mydomain.com/wp-admin/install.php` and follow the on-screen instructions. If you installed your WordPress files to a different location, head there, but make sure you append `/wp-admin/install.php` to the end. It couldn't be more simple to get up and running.

From there, you can customize the themes and modules of your WordPress install to your hearts' content. If you'd like more information on what you can do with WordPress, [head to its website](#).

### 3.8.4 - Using SSL for Trusted Connections

#### About SSL Certificates

This step is also optional but it is highly recommended, especially for any sites that will require logins or access to potentially sensitive information.

SSL is a method for web browsers to encrypt connection data between the client (your computer) and the source (the server you're trying to access). SSL can be found in use all over the web, nearly anywhere you need to login with something. Any address reachable or shown as "HTTPS" indicates a site that is compatible with SSL. From a privacy and security standpoint, it is a best practice to use SSL wherever possible.

Perhaps the most substantial barrier to the adoption of SSL security to websites is the trust relationship it requires of your site. Currently, one can reap the secure benefits of SSL by using default "SSL certificates" that one can generate themselves on their server. However, in order to have an SSL certificate that provides trust -- trust that your web server is who it says it is -- an SSL certificate must come from an external source called a Certificate Authority. 99% of the time, these certificate authorities charge for SSL certificates, often an arm and a leg. If you end up using a self-generated certificate instead, browsers will pop up with messages like "Untrusted SSL Certificate" and advise that you not proceed. This is obviously not an ideal system but it is the one we are stuck with at the moment.

The summary of this story is that SSL is very important if you are going to be doing any logging-in or exchanges of sensitive information via your web server. Self-generated SSL certificates are just fine for personal use, as you can easily bypass the Untrusted SSL Certificate messages yourself and still be able to use the encryption features it provides. However, if you plan on offering any services whatsoever to other people, such as shared email hosting for your organization or a cloud platform for your family, it is advised that you purchase an SSL certificate for use in your website.

## Producing a Certificate Signing Request

Whether you are looking for a self-signed certificate OR looking to purchase one for general service, you will need to generate a Certificate Signing Request (CSR). This is a file that will contain all the data a Certificate Authority needs to create our personalized certificate. First, we have to create a private key for our server to generate these requests with.

Note that if you already have a private key on your server for certificate requests or generation (generated for Postfix, for example) then you do not need to create another.

```
openssl genrsa -des3 -out server.key 1024
```

Make sure you save this file in a very safe location, as you should keep it for future certificate requests.

Now comes the time for our Certificate Signing Request:

```
openssl req -new -key server.key -out server.csr
```

OpenSSL will ask you several questions at this point, which should be tailored to your situation. These questions will include:

```
Country Name (2 letter code) [GB]:
State or Province Name (full name) [Berkshire]:
Locality Name (eg, city) [Newbury]:
Organization Name (eg, company) [My Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

You will need to set the "Common Name" as the fully-qualified name of the domain you wish to secure. If you wish to secure the base of your website located at <http://mydomain.com>, you can simply enter "mydomain.com." However if you want to

secure a subdomain like myblog.mydomain.com, you would need to enter "myblog.mydomain.com" here. You will need different certificates for different subdomains in the majority of cases. If you are obtaining a wildcard certificate (and you know what that is), you can enter "\*.mydomain.com" here.

Once you've answered the above questions, your CSR will be generated. At this point, you can either self-sign it for your own use, or you can send it to a Certificate Authority to purchase a certificate.

## Option 1 - Creating a Self-Signed Certificate

With your server key and CSR in hand, run the following to generate a certificate. It's generally a good idea to set a time limit on them and renew them after a certain period. This command will set it to expire after one year.

```
openssl x509 -req -days 365 -in server.csr -signkey server.key  
-out server.crt
```

And that's it! You can now dispose of the CSR file, but keep the CRT and the KEY in very safe places. We will use these two files in our Apache installation.

## Option 2 - Obtaining a Certificate from a Certificate Authority

To purchase a valid SSL certificate that is signed by a Certificate Authority like Thawte, Verisign or Comodo, you can go online and provide them with your generated CSR. A good place to do this is on [NameCheap](#).

Choose the SSL certificate that meets your needs, then purchase it. You will be forwarded to a page where you can upload your CSR file and input some information. After a time, you will receive instructions on how to receive your certificate, based on the type of certificate you ordered and the company that is providing it.

## Installing Your Certificate in Apache

Once you have your key and CRT files in hand, you are ready to install them in Apache. This must be done in a special VirtualHost file for your SSL-enabled host. Copy your default host file in `/etc/apache2/sites-available`, and name it something like `$name-ssl`. See the example below for the required lines:

```
<VirtualHost *:443>
    DocumentRoot /www/example1
    ServerName example.com
    ServerAlias www.example.com

    SSLEngine On

    SSLCertificateFile /path/to/certificate/file.crt
    SSLCertificateKeyFile /path/to/certificate/keyfile.key
    SSLCertificateChainFile /path/to/certificate/chainfile

</VirtualHost>
```

1. Note that the port number is 443, instead of 80 here. This tells Apache that this site will be provided on port 443, the standard for HTTPS connections.
2. `SSLEngine` must be set ON in order for Apache to serve the site via HTTPS.
3. `SSLCertificateFile` and `SSLCertificateKeyFile` are mandatory. Put the location of your CRT and KEY files here.
4. `SSLCertificateChainFile` is ONLY required if you were specifically given a chainfile from your Certificate Authority. If you were not given one, or you are using a self-signed certificate, do not include this line.

Also, don't forget to enable the Apache ssl module by running:

```
sudo a2enmod ssl
```

Once this is done, enable any new VirtualHost files you created via `sudo a2ensite $name-ssl` and reload your configuration with `sudo service apache2 reload`. Fire up your browser of choice and head to `https://mydomain.com` and enjoy your encrypted connection!

## 3.9. Your Own “Cloud”: Files, Calendar and Contacts

---

### 3.9.1 - What is ownCloud?

ownCloud is a framework for personal cloud services that you can run on any server, for work or personal use. In plain English, it gives you many of the same services that platforms like Google can provide for you on a daily basis. But, as with everything else in this guide, you get the benefit of assuring your own data and full ownership as well.

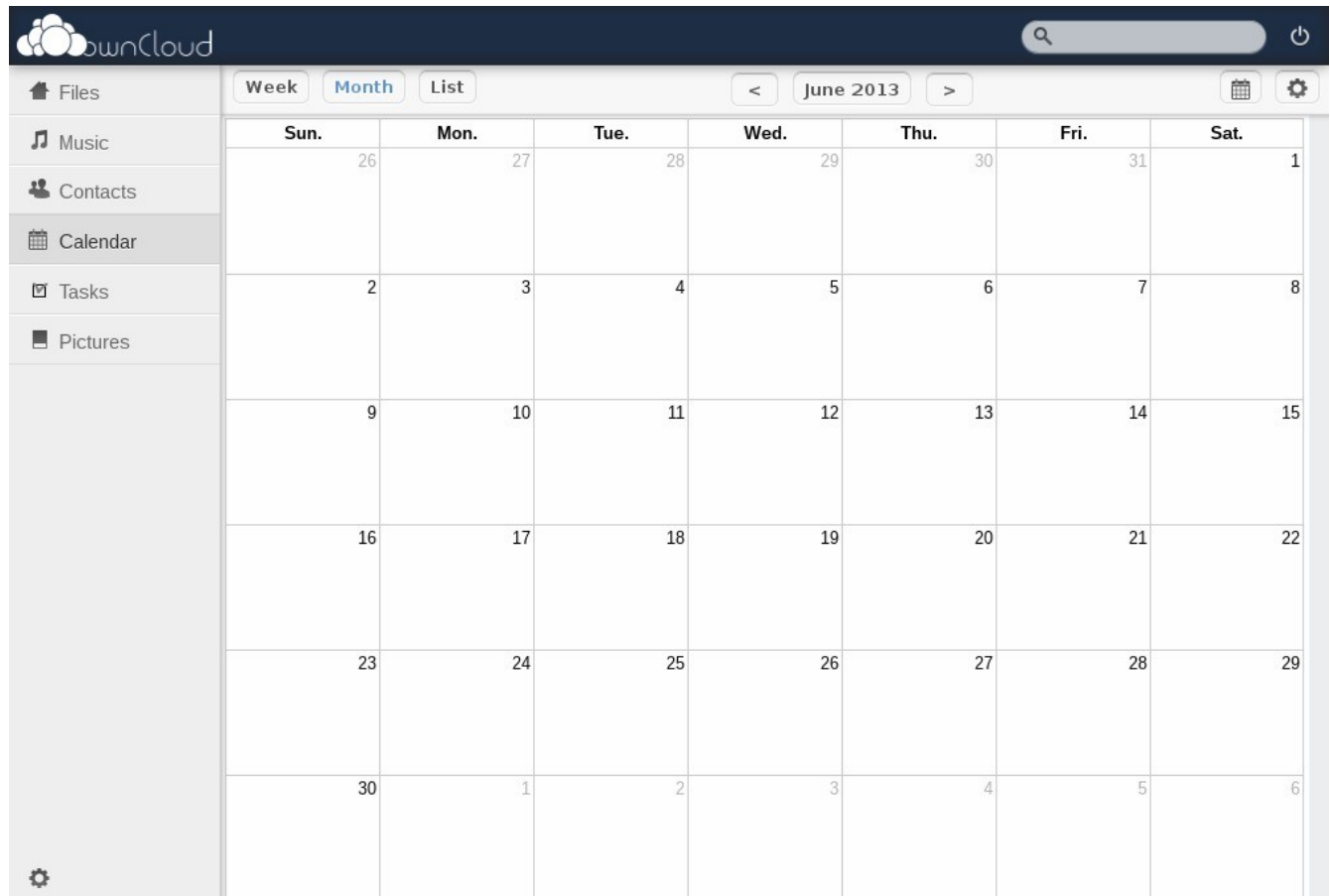
ownCloud has many features, as well as a plugin system that allows even more to be added externally. Here is a brief summary of its core functionality:

- **Files** - ownCloud can host your files for you, much in the same way you would do with your Google Drives or Google Docs. You can then access them on any web-enabled device, anywhere around the world, just like your Google Drive. The only difference is that you cannot -yet- edit rich text documents or spreadsheets like you can with Google Drive. That being said, it has built-in document readers (including for PDF) which makes it a decent everyday alternative to the file storage features of Google Drive.
- **Music** - You can also host your music library with ownCloud and be able to play it from anywhere in the world. This is a major plus if you have a phone or music device with limited storage space, and you cannot put everything you want on the device at once. Or, if you are at a friend's home and you want to show him some of your new tracks. ownCloud's built in music player is fast and easy to use.
- **Photos** - Just like you can host your music and files, you can also store your photos, eliminating the need for external services like Flickr or Picasa. Set up galleries and share them with others via the built-in interface.
- **Contacts** - One of my personal favourite features of ownCloud is its contact storage system. First, you can set up and access your email/phone contacts easily from the web interface. But where it really shines is its CardDAV syncing system. You can set it up to sync with your other computers and devices whenever a contact is added on any of them, easily replacing the contact sync features of Gmail or Apple's iCloud.
- **Calendar** - With the built-in calendar system you can view your schedule locally or remotely. But where it gets really useful - like with Contacts - is the sync capability. You can sync your devices' calendars with your ownCloud calendar via CalDAV, and



whenever you add or modify an event from one of these devices, all the others will update seamlessly. This replaces Google Calendar or Apple's iCloud.

- **Tasks** - This feature isn't as cleanly implemented as it should be (yet), but you can easily keep track of your tasks via the ownCloud's easy-to-use web interface.



### 3.9.2 - Installing ownCloud

Download the latest ownCloud source from its [website here](#). Make sure you choose the most recent branch of code available - at the time of writing that is 4.5.



Note that you will need to have Apache installed and configured properly to use ownCloud. Don't skip that guide! We cover Apache installation in guide 3.8.

Extract the package to the path of your choice. If this is the only web service you will use on your server, that path would be `/var/www/`; otherwise, you should extract it to `/usr/share/`.

```
tar -xvjf owncloud-*.tar.bz2
cp -r owncloud /path/of/your/choice
```

Next, go into that directory and make sure that certain critical files have the correct permissions. Change the owner of "apps" "config" and "data" folders, and all of their contents, to that of the webserver application. This guide will assume that you are running Apache as your webserver, like we established earlier in the guide, so that user is named `www-data`.

```
cd /usr/share/owncloud
mkdir data
chown -R www-data:www-data /usr/share/owncloud/install/apps
chown -R www-data:www-data /usr/share/owncloud/install/config
chown -R www-data:www-data /usr/share/owncloud/data
```

Next we will set up our Apache VirtualHost file for this service. This guide will show how to make a configuration that works over HTTPS, and automatically redirects any HTTP connections to HTTPS. Create a new file in `/etc/apache2/sites-available` named `owncloud` that resembles something like this:

```
<VirtualHost *:80>
    ServerName subdomain.mydomain.com
    DocumentRoot /usr/share/owncloud
    RewriteEngine On
    RewriteCond %{SERVER_PORT} !^443$
    RewriteRule ^.*$ https://%{SERVER_NAME}%{REQUEST_URI} [L,R]
</VirtualHost>
```

Now, create one named `owncloud-ssl` in the same folder, replacing the SSL certificate location and information where necessary.

```
<VirtualHost *:443>
    ServerName remote.jcook.cc
    DocumentRoot /usr/share/owncloud
    <Directory /usr/share/owncloud>
        AllowOverride All
    </Directory>
    SSLEngine On
    SSLCertificateFile      /etc/ssl/certs/mycertificate.crt
    SSLCertificateKeyFile   /etc/ssl/private/myprivatekey.key

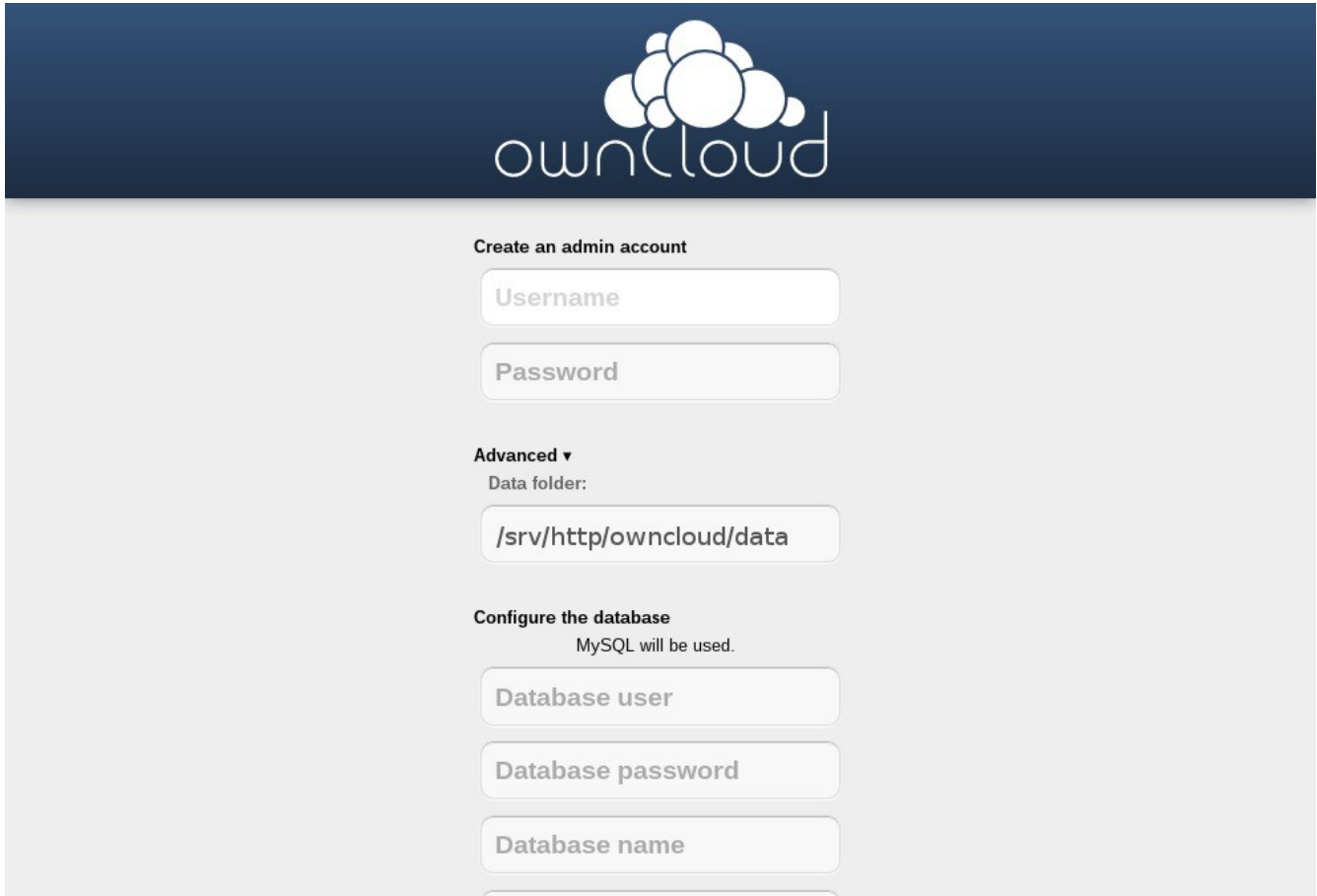
</VirtualHost>
```

Make sure you include the `AllowOverride All` in there; that will allow ownCloud to set its own custom parameters for security purposes.

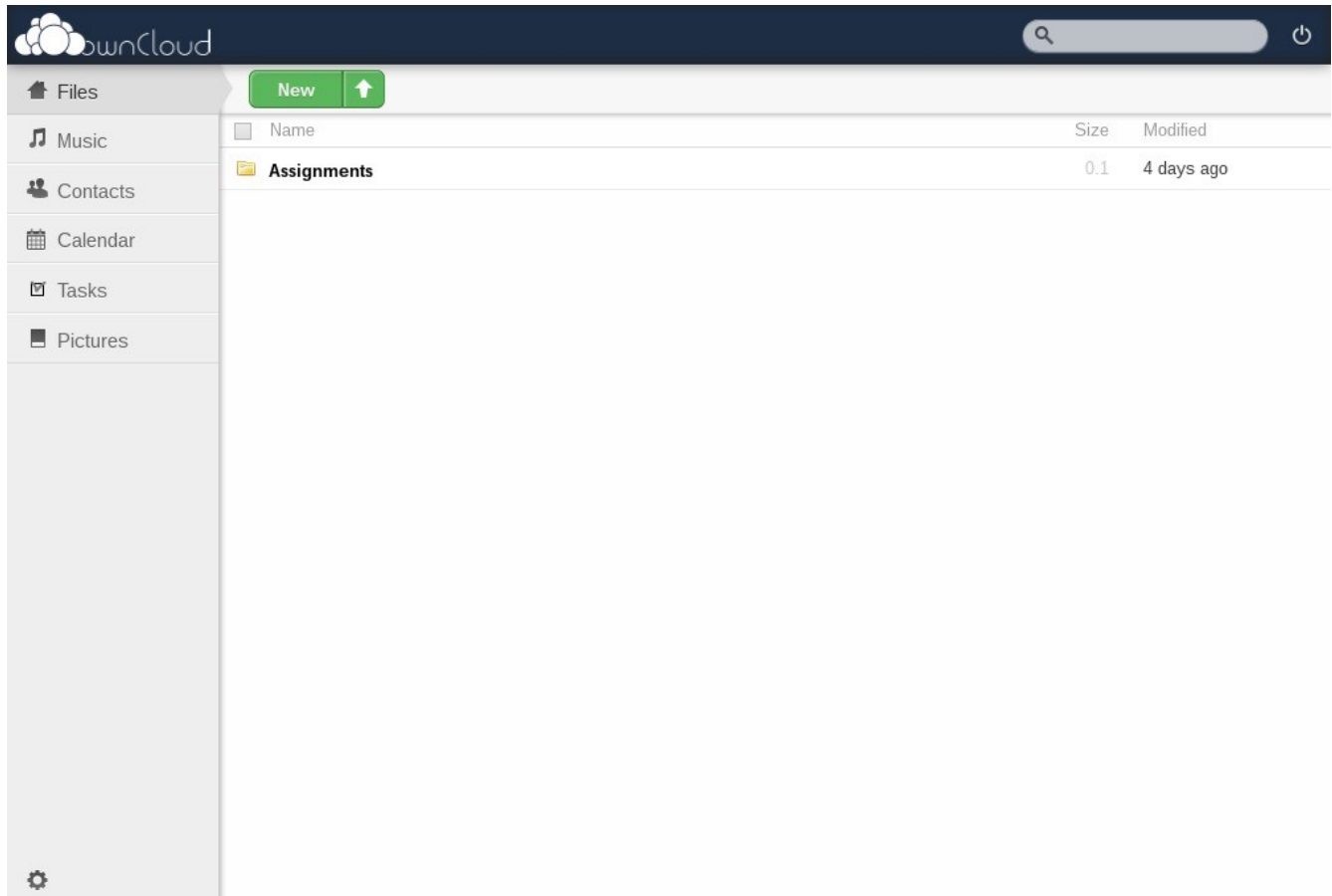
Save these and activate them in Apache, then we are ready to go!

```
sudo a2ensite {owncloud,owncloud-ssl}
sudo service apache2 reload
```

Open up your browser and navigate to the server name that you set up earlier. You will be guided through an installation wizard that will set up a database and administrative user. For more information about this, refer to guide chapter 3.8.

The image shows the ownCloud installation wizard interface. At the top, there is a dark blue header with the ownCloud logo, which consists of a cluster of white circles of varying sizes above the text "owncloud" in a white, lowercase, sans-serif font. Below the header, the main content area has a light gray background. It features several sections for configuration: 1. "Create an admin account" with input fields for "Username" and "Password". 2. "Advanced" section, indicated by a small downward arrow, containing a "Data folder:" label and an input field with the path "/srv/http/owncloud/data". 3. "Configure the database" section, which includes the text "MySQL will be used." and three input fields labeled "Database user", "Database password", and "Database name". The fields are arranged vertically and have a clean, modern design with rounded corners and light gray borders.

Once ownCloud is properly set up, you will see its main screen which is easily identifiable. Click the gear icon and you will eventually find the Admin screen, where you can personalize more about your ownCloud implementation.



And that's it! You can use its intuitive interface to store and share your files, set up calendar appointments, and organize the media. Have fun with your own personal cloud!

### 3.9.3 - Setting up Contacts, Calendar and File Sync

As the contacts and calendar features of ownCloud are so helpful, this guide will also explain how to synchronize them with your personal devices.

#### Thunderbird: Contact and Calendar Sync

To synchronize your contacts with Thunderbird:

1. Go to your ownCloud web interface and click Contacts. From there, click the small gear logo underneath the contact list.
2. Click "More..." and select "Show CardDAV Link" next to the address book you want to sync. Copy the address that comes up.
3. Download and install the [SoGO Connector Thunderbird Plugin](#) in Thunderbird.
4. Open up Thunderbird and click "Address Book."
5. In the Address Book, click File > New > Remote Address Book. Set the URL as the one that you copied from ownCloud. Give it a name as well. If you do not want changes made in Thunderbird to be synced back to your ownCloud server, then choose "Read Only."
6. Click OK, then right-click the address book and choose Synchronize. (You may need to close and re-open the address book before this will work.)

To synchronize your calendar with Thunderbird:

1. Go to your ownCloud web interface and click Calendar. From there, click the calendar logo in the upper right.
2. Choose the calendar you want to sync, then click "CalDAV Link" and copy the link.
3. In Thunderbird, install the Lightning calendar addon by going to Tools > Addons then searching for Lightning and installing it.
4. Restart Thunderbird and click the Calendar icon that appears in the upper right.
5. Right-click in the Calendar field and choose "New Calendar..."
6. In the window that pops up, choose "On the Network" and click Next. Choose CalDAV and place the calendar URL you copied from ownCloud in the space provided. Click Next.

7. Choose a name and colour for your calendar, then click Next. Your calendar will automatically be synchronized.

## **ownCloud Client: File Sync for Desktops**

To keep your ownCloud folder synchronized with a folder on your computer, the easiest way is to install the custom ownCloud client for desktop. There are versions for Windows, Mac OS X and Linux. Go to [the sync clients page](#), download the installation package, and follow the easy-to-use wizard to get it set up.

## **Android: Contact, Calendar and File Sync**

On Android, you must download/purchase a couple of applications in order to sync your contacts and calendars.

For contact sync, download [CardDAV Sync](#) from the Play Store. From here, you can add a Sync account from your Preferences application, just like if you were to add a Google account to your phone. For calendar sync, download [CalDAV Sync](#) which is made by the same developer and can be configured similarly.

For file synchronization, there is the [ownCloud app](#) that will enable cloud sync between your phone and your ownCloud server. Or, you can use any "cloud sync" app on the Android that supports the WebDAV format. Just go into ownCloud, click Settings > Personal, then copy the WebDAV link into the application.

### **3.9.4 - Further Reading**

- [ownCloud Documentation Centre](#)

## 3.10. Security: Firewalling and Threat Detection

---

### 3.10.1 - *ufw, the Uncomplicated Firewall*

One of Linux's most commonly used firewall systems is `iptables`. `Iptables` is an extremely customizable and extensible firewalling solution, however it is very complicated to set up and maintain on its own. Luckily we have `ufw`. `ufw` operates by establishing certain rules in its own front end, then translating those rules into the many lines that `iptables` can understand and execute.

Install `ufw`:

```
sudo apt-get install ufw
```

We will set our firewall to deny connections that we have not explicitly granted by default. To do this, run:

```
sudo ufw default deny
```

At this point, we can enable our firewall with:

```
sudo ufw enable
```

Now it will be up to us to set specific rules (and open ports) based on the apps/servers we are running. This goes for anything operating off of this server or any other client on the internal network that requires open ports.

### 3.10.2 - *Setting ufw Rules*

To allow traffic through our firewall, we will need to allow ports through it. We can do that with:

```
sudo ufw allow $xxxx
```

This will allow any system externally to use port "xxxx" on your server. This may be a good



option for setting up indiscriminate servers like for email or web hosting, but what if you want to only offer services to your internal network? For example, you might host a Samba server to share and edit files on the network, but you might not necessarily want this open to the internet, even if it is password-protected. Then the following rule is for you:

```
sudo ufw allow from 192.168.0.0/24 to any port $xxxx
```

This allows any system on your network (that has an address in the range of 192.168.0.0) to access port "xxxx" on your server.

Notice that UFW can also recognize a certain number of applications and server names instead of just the port number. You could run:

```
sudo ufw allow Apache
```

... and UFW would open port 80 on your server to the Internet. Port 80 is the web port that Apache accepts connections on.

To list your rules, run `sudo ufw status numbered` and you will get a numbered list of rules that are currently active. To delete a rule you've already set up, grab its number from that list and run `sudo ufw delete $xx`.

This guide explains how to set up several different services that, depending on your usage, you may want to open up to your internal network or the internet. Remember that the ports have to be open before you will be able to use them! Here is a list of the common applications and ports you might want to allow through.

- **Remote Connect:** port 22 for SSH, port 5900 for VNC
- **Mail:** port 143 for IMAP, port 110 for POP; port 25 for SMTP, port 587 for SMTP submission (optional)
- **Web:** port 80 for standard HTTP, port 443 for SSL HTTPS, port 3306 for MySQL
- **Network Controller:** ports 53 and 953 for DNS, ports 67 & 68 for DHCP, port 5351 for NAT
- **File Sharing:** port 21 for FTP, ports 137-139 and 445 for Samba, port 69 for TFTP, port 192 for AFP (Apple Filesharing Protocol), port 2049 for NFS (Linux Filesharing)
- **Windows Services:** port 139 for NetBIOS, port 161 for SNMP
- **Media Streaming:** port 3689 for DAAP (Apple/iTunes), ports 1900 and 5000 for uPnP

### 3.10.3 - SSH Tunnelling: Maintain Secure Access through a Closed Firewall

For applications you don't want to allow through to the Internet (if you think you are going to rarely use them away from home, or if you have significant security concerns), but you still might want to use them someday, it's good practice to use them over an SSH tunnel. You can create SSH tunnels with the following format, replacing the values where necessary:

```
ssh -f -L $localport:localhost:$remoteport $remotehost -N
```

The local port should be a port that is not already in use on your client computer. `localhost` can be left alone; this creates the tunnel to your client computer. `remoteport` refers to the port for whatever service you want to tunnel through. And of course `remotehost` is the address of your server on the Internet. So, for example, if you set up the following...

```
ssh -f -L 9876:localhost:5900 server.mydomain.com -N
```

... this will create an SSH tunnel for Minecraft on my computer. Simply connect Minecraft to a server located at `localhost:9876` and you can use it via a remote connection, as if you were just connected to the local network.

### 3.10.4 - Setting up fail2ban

Our firewall is in place, which will go a long way to helping secure our system from most attack attempts. However we will go a step further by using `fail2ban`. Fail2ban monitors the logs of network-capable applications for entry attempts and repeated failures, and promptly bans the associated IP addresses for a determined amount of time. This can help dissuade and eliminate the threat posed by certain bots that like to roam the internet, testing many different attack strategies at once to try and find one that sticks. It also helps stop some DDoS attempts, which are commonly used to bring down websites and other services.

Install fail2ban with:

```
sudo apt-get install fail2ban
```

Make a copy of the configuration template to the one we will actually be working from:

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Open up `/etc/fail2ban/jail.local` in your text editor and modify the following fields:

- **ignoreip:** set this to your internal network's subnet (like 192.168.0.0/24). This will keep fail2ban from blocking you if you trip its conditions while testing your services.
- **bantime:** Like it says on the tin, this is the amount of time the offending IPs are banned for (in seconds).
- **maxretry:** By default, the amount of failed attempts that should be allowed before an IP is banned.

Further down in the file, you will find a section for "Action Shortcuts." The line for default action begins with "action," and can be set here. There are three options when triggered, explained above: ban only, ban and send an email with the IP information, or ban and send an email with IP AND relevant log information.

The next section, "Jails," deals with the services we want to monitor. The entries you make in this section will depend on what services you have enabled. There are sections installed by default for ssh, Apache, ftp, postfix, etc. For some services, there are multiple jails, each that monitor for different circumstances. For example, "apache" monitors authorization attempts to your website, while "apache-php" monitors repeated failures in access to PHP files, which can often signify someone fishing for a way into your site's configuration. Make sure you select any jail that you feel you will need based on your setup.

To enable a jail, uncomment every line within it after the "[xxxx]" section, then set enabled to equal "true".

Once you have completed the configuration, you can start fail2ban with `sudo service fail2ban start`. It will immediately begin monitoring the selected services and banning repeat offenders accordingly.



With some services, errors can come up innocently yet frequently. If you create a broken link to a PHP page on your website, for example, and you have the apache-php jail enabled, you might be sending people to an error screen that can ban them if they try to refresh it too much. Keep this in mind when enabling jails AND when choosing a ban time.

### 3.10.5 - More Security Tips

- **SSH:** Make sure you've disabled root logins as well as password logins for SSH, and are only using key-based logins if at all possible. Remember to keep your keys safe on your devices!
- **Mail:** Remember to set your `smtpd_recipient_restrictions` in Postfix, and to make `ssl = required` in Dovecot.
- **Apache/ownCloud:** Any websites set up that handle password authentication or the transmission of even *remotely*-sensitive data should have HTTPS enforced in the settings.
- **File Sharing and Media Streaming:** Set permissions where possible so that only your authorized network users have read/write privileges. Block services like Samba, uPnP and AFP from being used outside your network by blocking their ports at the firewall.
- **Backups:** Encrypt any backups that are stored on your server for an extra level of protection.
- **In General:** It's better to use SSH tunnels for applications you only use remotely on an infrequent basis than to leave them open and forget about them!

Instructions for how to enable these tips can be found in their respective sections of this guide.

### 3.10.6 - Further Reading

- [UFW Questions and Answers - Ubuntu Forums](#)
- [Fail2ban Wiki](#)

## 3.11. Managing and Streaming Your Media

---

### 3.11.1 - Setup File Shares via Samba and NFS

The first step to setting up a file server, whether its for your local network or for remote access, is to decide upon a method for sharing that works with your desired configuration. This guide will explain three different systems, each easy to set up but used for different purposes. You can set up all three, or any combination thereof.

#### Samba

Samba is a file sharing server that allows your Linux server to interact with Windows clients on your network. It also easily works with Mac OS X clients. If your home network includes any devices that do not run Linux, and you want those devices to be able to interact with your files stored on the server, it is usually a good idea to set up Samba.

All you need to set up Samba is `sudo apt-get install samba`. After this, you will be ready to add a new share.

Open up `/etc/samba/smb.conf` in your text editor and scroll to the bottom of the file. You will want to add a section that looks like this, changing the fields where appropriate:

```
[share]
    comment = My File Server Title
    path = /path/to/my/shared/folder
    browsable = yes
    guest ok = yes
    read only = no
    create mask = 0755
```

The `guest ok` field above will change if people can use your file server without logging in with a password. The `read only` field will change if someone logged in to your server is able to change the files at all, or just to read from them.

Once this is complete, you merely need to restart your Samba server:

```
sudo service smbd restart
```

Now, to connect, open up your file browser on a computer connected to your network.

- Windows: In the Address Bar of your file browser, enter `\\$servername\share`. If you want to mount this share permanently like a hard drive, right-click Computer and choose "Map Network Drive." Put the above address in as the folder, and choose a drive letter.
- Mac OS X: In Finder, click Go > Connect To Server. Insert the address "smb://\$servername/share".



It is not a good idea to open your Samba server to the world. For sharing with others, use FTP or a separately-installed service like ownCloud. Use a firewall like `ufw` to block Samba's ports externally, or to only allow it on your local network. The ports used for Samba are 139 and 445.

The easiest way to improve the security of this setup is to require users to log into your server via user accounts. You can easily do this via PAM, which is the software that runs your Linux server's user accounts and logins. To do this, run `sudo apt-get install libpam-smbpass`. Then go back into your Samba configuration file and set `guest ok` to equal "no". Restart your Samba server with `sudo service smbd restart`.

With this, you can restrict your file access to only users that have accounts on your server.

## NFS

NFS is a network filesharing system designed for Linux systems. It is a faster and easier option than Samba if you are only planning to use your fileserver with Linux-based computers.

To install NFS, run `sudo apt-get install nfs-kernel-server`. Then, to add a new share, edit the `/etc/exports` file, and add lines based on the following configuration.

```
/path/to/shared/folder *(ro,sync,no_root_squash)
```

The first 'ro' indicates if this share should be read-only or writable to clients that connect to it. To make it writable, replace 'ro' with 'rw'. If you want to restrict this share to be available only to specific computers on your network, replace the '\*' with those computer hostnames, IP addresses or IP range/subnet.

After adding your shares, start your server with `sudo service nfs-kernel-server restart`.

To connect to these systems from your Linux computer, go to the Terminal and run `sudo mount $servername:/path/to/shared/folder /path/to/local/mount`. You will need to set up a folder on the computer to act as the local mount point. After this, you can go to that folder path and use it, just as if it was a local folder.



Much like Samba, you shouldn't open your NFS server to the world. For sharing with others, use FTP or a separately-installed service like ownCloud. Use a firewall like ufw to block the NFS ports externally, or to only allow it on your local network. NFS uses port 2049 for its connections.

### 3.11.2 - Stream Music/Photos/Video via uPnP

Once we have our file servers set up, that's all well and good, but it does not let us seamlessly stream our content. That is one of the great benefits of having a server act as a NAS (network attached storage) device: being able to stream your media from various devices around your home. uPnP is one of the mechanisms that can be used to achieve this.

With it, you can seamlessly stream your music, photos or video with different platforms.

This guide will use a simple uPnP server called minidlna. It can stream to uPnP or DLNA compatible clients.

Install minidlna with `sudo apt-get install minidlna`. Configuration can be adjusted by editing the file `/etc/minidlna.conf`. The important lines to change based on your configuration are as follows:

1. **network\_interface** - If you have multiple network interfaces on your device, make sure they are listed here. Or, only list the network interfaces you want to serve. If you have one dedicated to the internal network and one facing your modem, you can easily prevent external access this way.
2. **media\_dir** - This will point your minidlna server to the folders containing the media you want to serve. An example: ``media_dir=A,/home/user/music`` will set minidlna to share the 'a'udio listed in these folders. V is used for video and P is used for photos. You can simply put ``media_dir=/home/user/folder`` if you have one folder with multiple types of media to stream.
3. **friendly\_name** - The name of your server that will be broadcast to clients.
4. **album\_art\_names** - If you have a specific naming convention for the album art in your music library, like "Cover.jpg", put it here. minidlna will set these files apart and use them for album covers in the library view.

With this, start your minidlna instance with:

```
sudo service minidlna restart
```

... then connect with your client of choice, and enjoy the streaming experience!



Remember to block the uPnP port to the outside world via your firewall if you don't want anyone and everyone to have access to your media collection! uPnP uses ports 1900 and 2869.



### 3.11.3 - Stream to your Apple Devices with DAAP

If you have an abundance of Apple devices in your home, or are just attached to your iTunes library more than anything else, you can use DAAP streaming instead of (or in addition to) uPnP. The DAAP setup is very similar to that of uPnP, but it will instead allow you to stream directly to iTunes using its "Home Sharing" functionality. There are also DAAP clients for Windows, Linux and Android.

We will install a DAAP server called `forked-daapd`. Run `sudo apt-get install forked-daapd`. To configure, we will edit the file `/etc/forked-daapd.conf`. You will need to set the 'directories' line to match the path to your music folder. You also may want to change the 'name' line to match what you want to show to your clients.

After this, restart the server by running `sudo service forked-daapd restart`. Then fire up iTunes, enable Home Sharing, and your server will show up in the sidebar!



Remember to block the DAAP port to the outside world via your firewall if you don't want anyone and everyone to have access to your media collection! DAAP uses port 3689 for its connections.

### 3.11.4 - Further Reading

- [Securing a Samba Print and File Server - Ubuntu Documentation](#)
- [NFS HOWTO - Ubuntu Community Documentation](#)
- [minidlna - Ubuntu Community Documentation](#)
- [How to set up forked-daapd - Ubuntu Forums](#)

## 3.12. APPENDIX: Guide to Virtual Machines

---

### 3.12.1 – What are Virtual Machines?

A virtual machine is a simulation of an operating system that can run within a different operating environment. Rather than only booting an operating system natively, like we do every time we start up our computers, virtual machines make it possible to run different operating systems and supported applications natively from your computer.

Here are a few use cases for using a Virtual Machine:

- **You work frequently on applications that require Windows.** There are many special or proprietary apps that are only available on Windows for specific lines of work. You can install Linux as the principal operating system and run Windows via a virtual machine whenever you need to access this specific program.
- **You have a server, and you want to run multiple types of server architectures on the same machine simultaneously.** For example, you want to operate an instance of Ubuntu Server to provide your email and web hosting, but you want to use FreeNAS to provide your media hosting and filesharing services. You can do this very safely on one computer just by using virtual machines.
- **You want to practice working in a specific operating system or Linux distribution before switching over to it entirely.** Linux distro live CDs can be slow, and not necessarily indicative of the real user experience. Trying out an OS or distro in a virtual machine before you switch your whole computer is a great way to see what you might be getting yourself into.
- **You are a software developer, and you need to test your program on a different operating system, or under different configurations.** There is obviously no need to buy multiple computers to achieve this, when you can simply install the OS via a virtual machine. You can also use specifically old operating system images to test how your software reacts to these environments.

In order to use a virtual machine, your computer or server must have adequate processing power. It is advised that your computer have a 64-bit processor, with at least two cores. Intel processors can feature VT-x technology, which is highly recommended for running virtual machines.

### 3.12.2 – Install VirtualBox and Set Up a VM

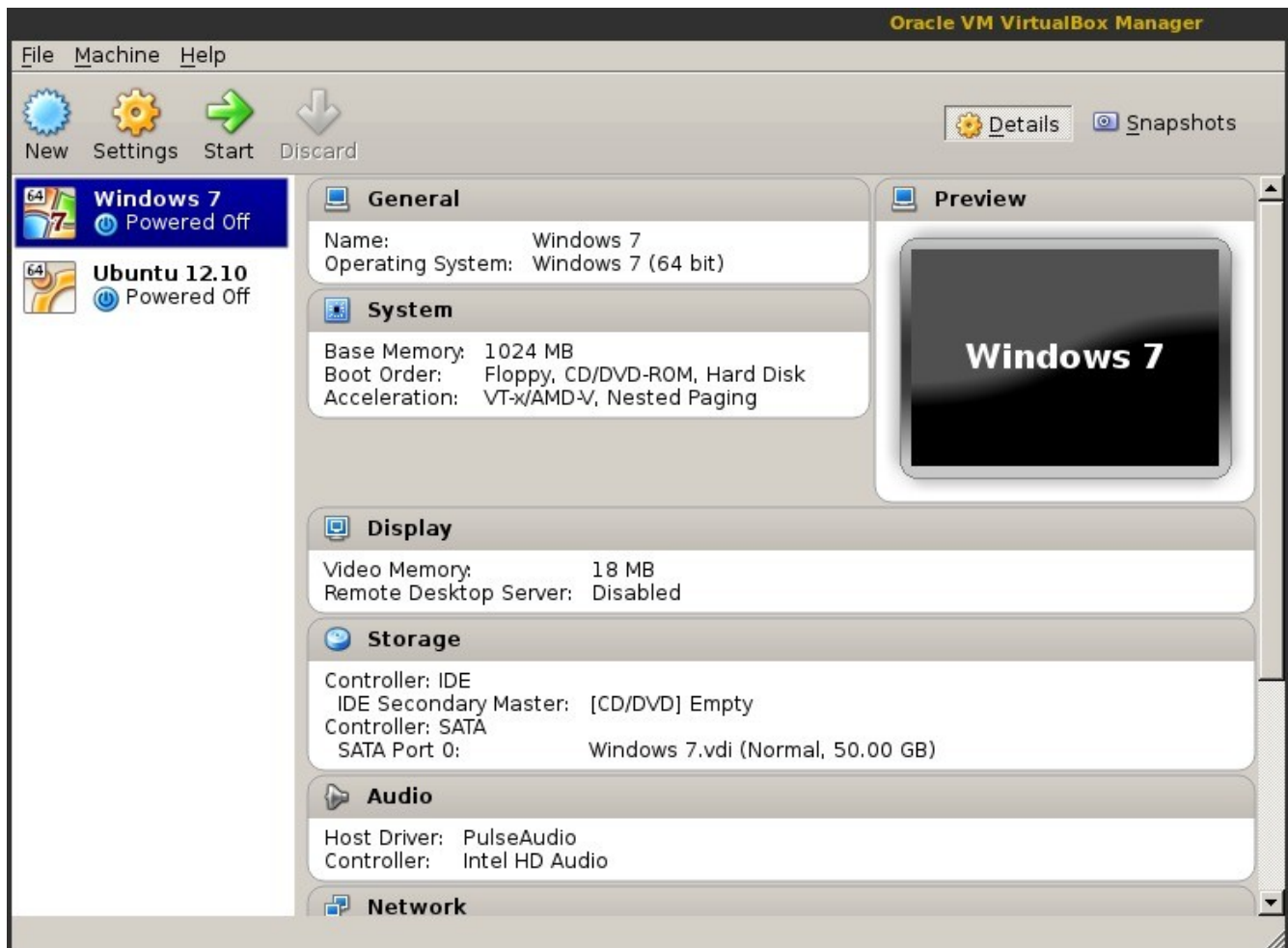
These instructions are for Oracle VirtualBox, an application that manages and runs virtual machines. It will show how to run VirtualBox via command-line (for Ubuntu Server) and graphical interface.

If your processor has special virtualization capability (required for 64-bit VMs), like Intel VT-x, you will need to enable this in your BIOS/UEFI configuration first. Look through your motherboard's manual for instructions on how to do this.

To install VirtualBox, run `sudo apt-get install dkms virtualbox`. This will install VirtualBox and the system to keep its required kernel modules up to date.

#### Via Ubuntu Desktop

Go to the sidebar and launch VirtualBox from the Search menu. You are presented with the main screen.

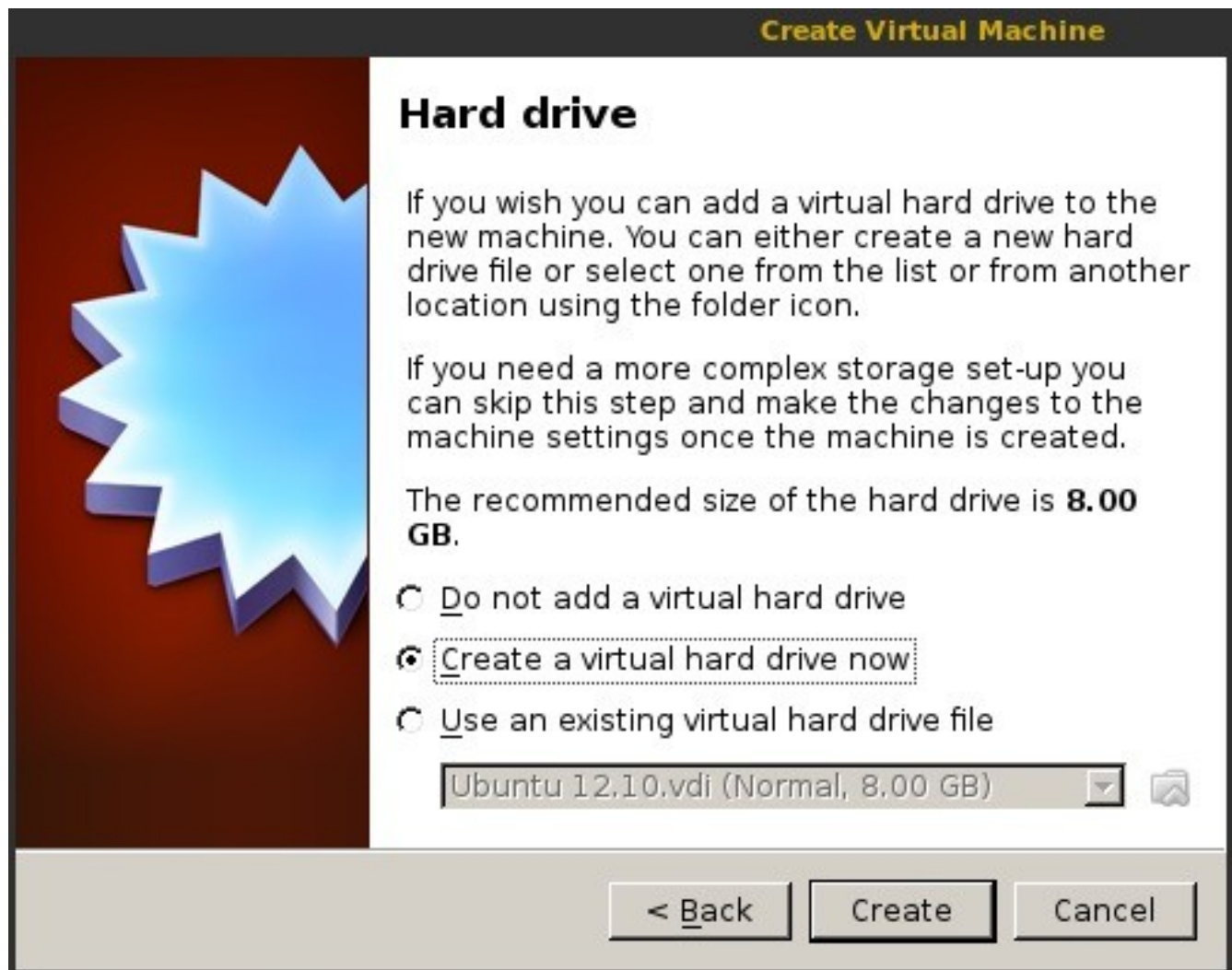


From here, you can see a list of your virtual machines in the left-hand side bar, as well as the controls for your VMs right above that.

To create a new virtual machine, click New, which will bring up the wizard. It will ask you to give a name to your virtual machine, to choose the operating system type and the version of your operating system.

Make sure that, if you wish to run a 64-bit operating system, you choose the 64-bit version of your OS displayed here.

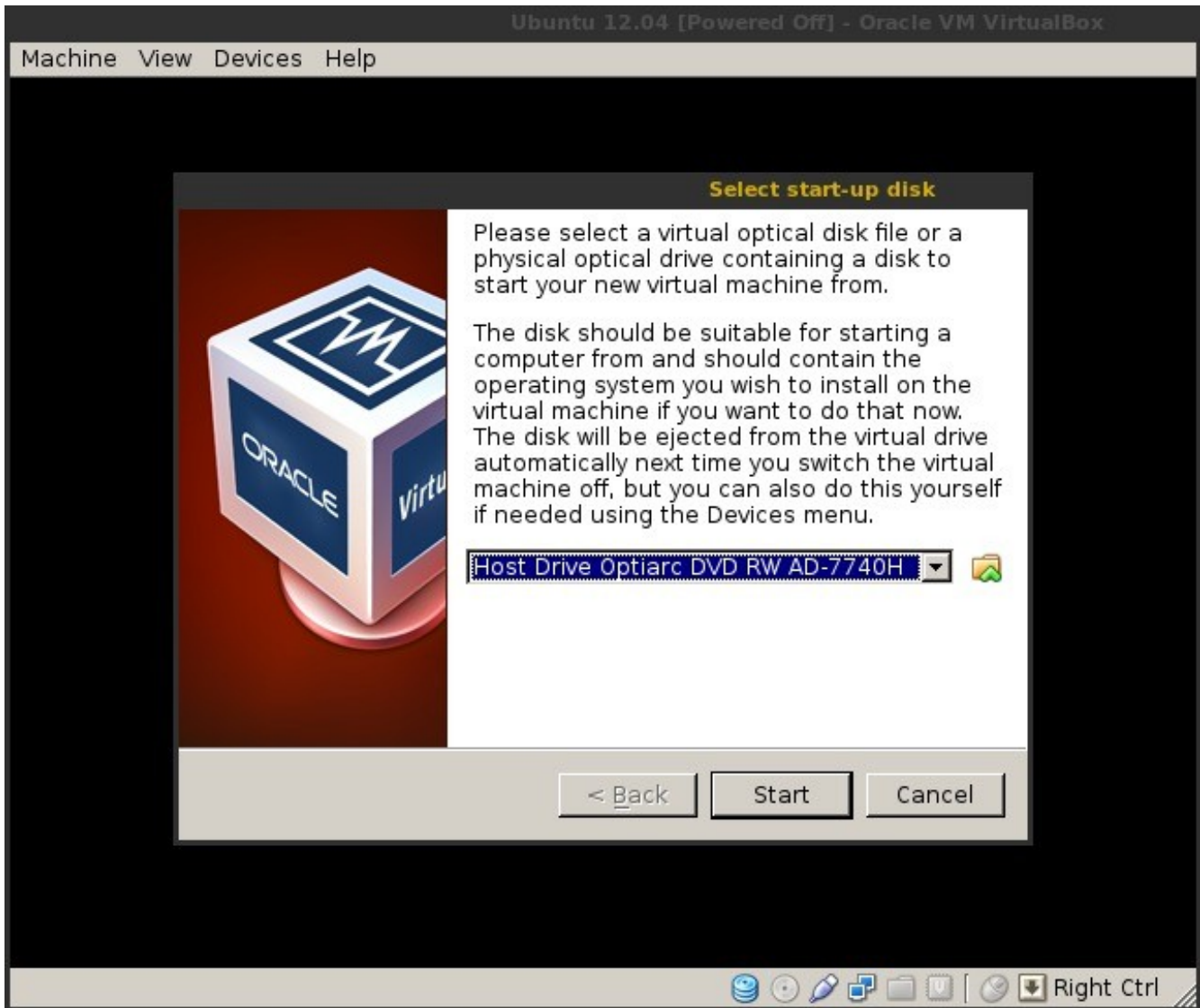
The next screen will allow you to choose the memory size of your virtual machine. For most Linux-based VMs, 512MB will suffice. For Windows-based VMs it may be a good idea to use 1024MB. This will of course depend on how much RAM your system has to begin with.



Now you can set the virtual hard drive space that your virtual machine will run from. This will create a file that acts as a container for everything held in your virtual machine. Click "Create" and choose "VDI (VirtualBox Disk Image)." The next screen will allow you to choose a dynamically-allocated image or a fixed-size image. Dynamically-allocated images are a good option, because you can set a maximum theoretical size for the image without actually taking up all of that disk space until your VM actually does so. If you choose fixed-size, a 50GB disk image (for example) would instantly take up 50GB of space on your disk, regardless if the VM is actually using that much space or not.

The next screen will allow you to set the size of your virtual disk. You obviously must set a size that will fit on your physical hard drive. Linux distributions (especially ones that do not host media files) do not require much space to operate; they will do fine with a range of 10 to 20GB. Larger operating systems like Windows will require a minimum of at least 25 to 50GB to operate.

Once you click "Create," your new VM will show up in the list. When you are ready, click "Start" above the list to begin the process of installing your operating system. A screen will come up that will allow you to choose an installation source.



From here, you can choose either your CD drive (if you have your OS' installation disc loaded), or an ISO file to install from a downloaded install image. Then you can follow the normal installation process for your chosen operating system.

## Via Ubuntu Server (Advanced)

You can set up a new virtual machine in VirtualBox using the command line. Here is an example command and its important features:

```
VBoxManage createvm --name "Ubuntu 11.04 Server" --register
```

This creates our virtual machine, named "Ubuntu 11.04 Server," and registers it with VirtualBox.

```
VBoxManage modifyvm "Ubuntu 11.04 Server" --memory 512 --acpi on  
--boot1 dvd --nic1 bridged --bridgeadapter1 eth0
```

This sets our VM up with 512MB of RAM space, enables ACPI support, sets the machine to look for a DVD to boot from first before anything else, and sets up a network interface that bridges to our own, so that we can use the Internet from our virtual machine. If you are using a wireless network card instead of an ethernet card, make sure you change `eth0` to `wlan0`.

```
VBoxManage createhd --filename Ubuntu_11_04_Server.vdi --size 10000
```

This creates a virtual hard drive file named "Ubuntu\_11\_04\_Server.vdi", with a size of 10,000 MB (or 9.77GB).

```
VBoxManage storagectl "Ubuntu 11.04 Server" --name "SATA Controller"  
--add sata
```

This sets up a virtual SATA controller to connect our virtual hard drive.

```
VBoxManage storageattach "Ubuntu 11.04 Server" --storagectl "SATA  
Controller" --port 0 --device 0 --type hdd --medium  
Ubuntu_11_04_Server.vdi
```

This actually connects our virtual hard drive to our new virtual machine.

```
VBoxManage storageattach "Ubuntu 11.04 Server" --storagectl "SATA  
Controller" --port 1 --device 0 --type dvddrive --medium  
/home/ubuntu-11.04-server-amd64.iso
```

And finally, this connects a downloaded ISO install image on our hard drive, located at `/home/ubuntu-11.04-server-amd64.iso`, to our virtual machine, so that it will boot from it and install the operating system.

In order to run virtual machines "headlessly" - that is, without a direct monitor connection or a window environment so we can actually see it, we must enable a few extra features in VirtualBox. First, download [the VirtualBox extension pack](#) that corresponds to the version of VirtualBox that you are running. Next, install it in VirtualBox by running `sudo VBoxManage extpack install Oracle_VM_VirtualBox_Extension_Pack-*.vbox-extpack` from the folder you downloaded it to. Finally, run `sudo adduser $username vboxusers` with the appropriate username to give our user the ability to run the VM with these new features.

To actually run our virtual machine and begin the installation process, run `VBoxHeadless --startvm "Ubuntu 11.04 Server"`. Then, on a remote machine, connect to your server via RDP. You should be able to view the live contents of your virtual machine as it is running.

When you are done using your virtual machine, you should shut down the operating system it is running, just like you would a normal computer. If you need to kill it without a normal shutdown, run `VBoxManage controlvm "Ubuntu 11.04 Server" poweroff`.

### 3.12.3 - Further Reading

- [VirtualBox - Remote Virtual Machines](#)



## 3.13. APPENDIX: Guide to FreeNAS

---

### 3.13.1 – What is FreeNAS?

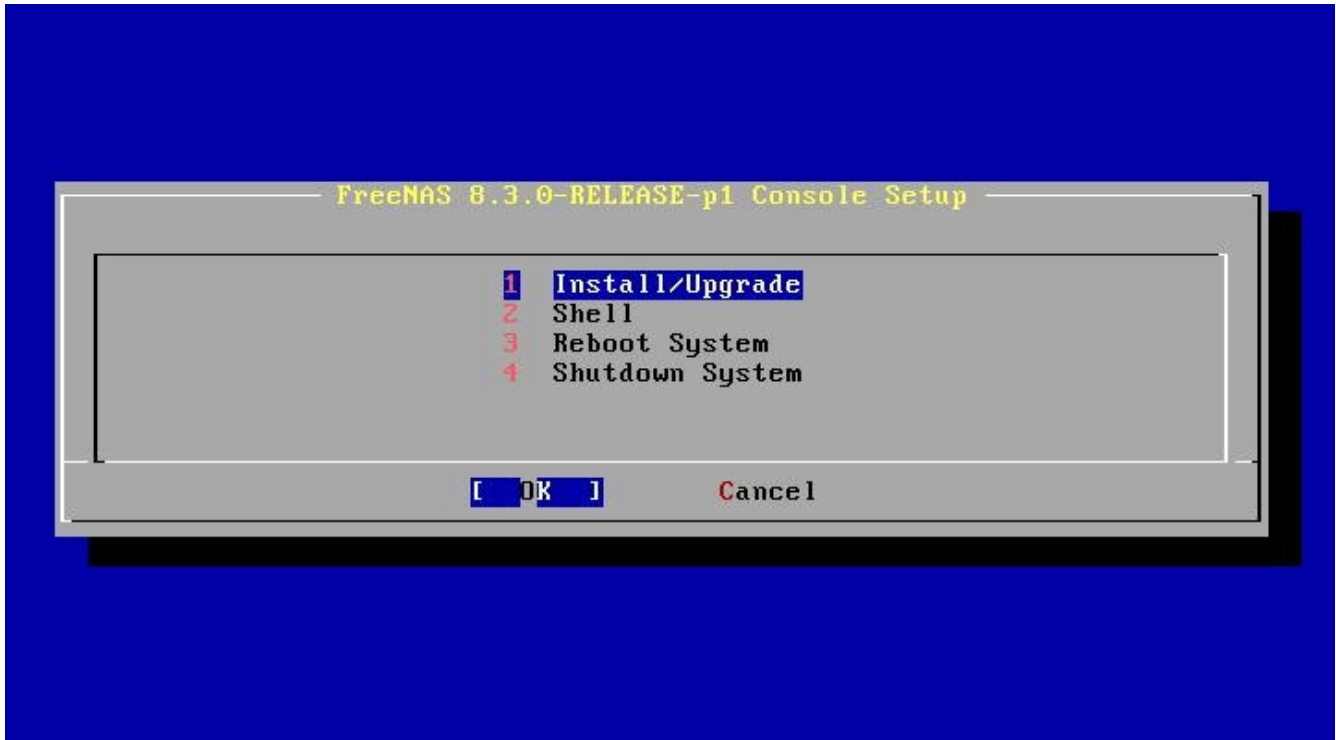
FreeNAS is a version of the BSD operating system that includes built-in and dedicated tools for operating file storage and media services. It is an excellent choice for those who wish to maximize their ability to oversee and control their media server, and retain a very easy-to-use and easy-to-setup interface.

FreeNAS is not just server software, like most other tools explained in this guide. It is a separate operating system. As such, it is designed to run on a dedicated NAS (network-attached storage) system or virtual machine. A NAS is used primarily to store a large amount of files or media at once, and to host high-capacity hard drives for this.

Oftentimes it is better to run your server under FreeNAS (or run a virtual machine with it) if you are planning on providing a decent amount of filesharing or media streaming services. It is also a good option if you will be using one server, but want to include sufficient controls over your media services and only will be running them on an internal network. By running FreeNAS on a virtual machine separate from your other web server software, you can ensure that external sources will not have the same access to this machine as your web server.

### 3.13.2 – Installing FreeNAS

FreeNAS is installed much like any other Linux-like operating system. Download the full version [ISO image](#) from the front page of the FreeNAS website, depending on your architecture. After it is downloaded, you can then burn it to a disc and boot your server from it to begin the installation process. If you want to run FreeNAS in a virtual machine, you can start your ISO with VirtualBox directly to install (check out chapter 3.12 for more information on virtual machines).



Once you boot from the CD/image, you are greeted with the lovely text-based installer. The instructions here will walk you through choosing the right disk partition. Installing FreeNAS is super easy, it's nearly a one-click installation. Once it is fully installed, it will let you know that it can reboot.

After the reboot, you are sent to its main menu.

```
FreeBSD/amd64 (freenas.local) (ttyv0)

Console setup
-----

1) Configure Network Interfaces
2) Configure Link Aggregation
3) Create VLAN Interface
4) Configure Default Route
5) Configure Static Routes
6) Configure DNS
7) Reset WebGUI login credentials
8) Reset to factory defaults
9) Shell
10) Reboot
11) Shutdown

You may try the following URLs to access the web user interface:
http://10.0.2.15/

Enter an option from 1-11: █
```

- Choose menu option 1 to give FreeNAS a custom IP address, or change the network interface it uses by default. FreeNAS will automatically try to autodetect your network settings and to receive an IP address via DHCP if you do not give it custom settings.
- Menu options 2-6 are used for those who have advanced network configurations and that need to supply link, routing or DNS information manually.
- Menu option 7 is used to reset the username or password used to log into the WebGUI. The WebGUI is the main way to add/remove shares and change settings for FreeNAS.
- Menu option 8 is used to reset your FreeNAS setup to its factory defaults; that is, to remove all of your custom configuration.
- Menu option 9 will bring you to a BSD command prompt, for advanced users only.
- And finally, options 10 and 11 will reboot or shutdown your FreeNAS system.



Note that if your network assigns IP addresses via DHCP, you will need to designate a static IP address for your FreeNAS implementation. You can read more about how to do this in chapter 3.6. Otherwise, your FreeNAS distribution is running as long as it is at this screen.

To begin setting up your fileshares and continue the configuration, fire up your web browser of choice and navigate to the URL that was listed on the screen. This will take you to the FreeNAS WebGUI.

The screenshot shows the FreeNAS WebGUI interface. At the top is the FreeNAS logo and a navigation bar with icons for System, Network, Storage, Sharing, Services, Account, Help, Log Out, and Alert. Below the navigation bar is a sidebar with a list of menu items: Account, System, Network, Storage, Sharing, Services, Display System Processes, Shell, Reboot, and Shutdown. The main content area displays the 'System Information' tab, which contains a table of system details.

System Information	
Hostname	omega-nas.geogaddi.home
Build	FreeNAS-8.2.0-RELEASE-p1-x64 (r11950)
Platform	Intel(R) Xeon(R) CPU E31220 @ 3.10GHz
Memory	1009MB
System Time	Wed Jan 30 20:00:57 EST 2013
Uptime	8:00PM up 26 mins, 0 users
Load Average	0.31, 0.17, 0.11
Connected through	omega-nas

At the bottom of the interface, there is a footer with the text 'FreeNAS™ © 2012 iXsystems, Inc.' and the iXsystems logo.

After you log in, this is the first screen you are greeted with, showing your basic system information. Menu options are listed along the left side and on the upper menu bar. You can customize various details about your admin account and set up users to connect to your NAS under the "Account" submenu. Under "System" you can configure your FreeNAS' details like time zone, email used for notifications, and other things. The "Network" submenu will

allow you to make any changes to your network connection and interfaces that you didn't make in the text-based menu earlier.

The next option, "Storage," will allow you to set up hard drive space to store the files and media that you want to serve with FreeNAS. To begin, click Storage > Volumes.

- If you want to use an existing partition on your hard drive, click "Import Volume." Set the name of the volume, then choose the disk/partition you want to use. Then select its filesystem type. Note that FreeNAS only supports filesystems of the UFS, NTFS (Windows), MSDOSFS (old Windows) or ext2fs (older Linux) types.
- If you want to create a new virtual hard drive to be stored on the disk, click Volume Manager. Choose a name for your new volume; it doesn't have to be very descriptive. "NAS" works just fine. Then choose "ZFS". Choose whether or not you wish to use full-disk encryption, though this is not recommended for large file or media servers. If you have more than one disk selected, you can set FreeNAS to mirror or stripe them using RAID (if you are not sure what RAID is, then [click here](#)).

Once your volume is set up, you are free to set up the sharing services you want to run on your FreeNAS server.

### **3.13.3 – Using FreeNAS Services**

FreeNAS supports a wide range of services to extend your server's use. We will begin with setting up two basic services: NFS file shares for Linux-based computers, and CIFS/Samba file shares for Windows-based computers. Note that you can also use Samba file sharing on Apple-based hardware, and it is much better than Apple's proprietary AFP service.

## NFS

To set up NFS file shares on your FreeNAS box, click Sharing > Unix (NFS) Shares, then click Add Unix (NFS) Share.

The screenshot shows the FreeNAS web interface. The 'Add Unix (NFS) Share' dialog box is open, displaying the following fields and options:

- Comment:** A text input field.
- Path:** A text input field with a 'Browse' button next to it.
- Authorized network or IP addresses:** A large text input field.
- All Directories:** A checkbox with an information icon.
- Read Only:** A checkbox with an information icon.
- Quiet:** A checkbox with an information icon.
- Maproot User:** A dropdown menu showing 'N/A' with an information icon.
- Maproot Group:** A dropdown menu showing 'N/A' with an information icon.
- Mapall User:** A dropdown menu showing 'N/A' with an information icon.
- Mapall Group:** A dropdown menu showing 'N/A' with an information icon.

The background interface shows the 'Sharing' menu expanded, with 'Unix (NFS) Shares' selected. The 'Add Unix (NFS) Share' option is visible under this menu.

1. Add a "comment" that can identify your share on some systems.
2. Choose the path for your share on the local drive. This should match the mount point path that you created during the Storage step above.
3. If you want to limit the share to only be accessible to a certain IP address or range, enter it here.
4. If you want to make your share read-only to all users, check the "Read Only" box.

Otherwise, permissions will default to the Unix file permissions that your files have on the server.

5. Set any of the other advanced permissions if need be, then click "OK" when done to create your new share.

To activate your newly-created share, click Services > Control Services, then toggle the On switch next to NFS. To connect to your NFS share on a Linux-based computer, run the following command with the appropriate values. Remember that you must set up a local folder to act as the placeholder when it is mounted.

```
sudo mount $ip-address:/path/to/mount /path/to/local/folder
```

## CIFS/Samba

To set up CIFS file shares on your FreeNAS box, click Sharing > Windows (CIFS) Shares, then click Add Windows (CIFS) Share.

1. Add a name and/or a "comment" that can identify your share on some systems.
2. Choose the path for your share on the local drive. This should match the mount point path that you created during the Storage step above.
3. If you want your share to be browsable by clients in Windows Explorer (which you probably do), check the "Browsable to Network Clients" box.
4. If you want to make your share available to guest users, i.e. users that do not need to log into your server with a username/password, check "Allow Guest Access." You can also check "Only Allow Guest Access" if you do not want people to be able to log in via a user account.
5. Set any of the other advanced permissions if need be, then click "OK" when done to create your new share.

To activate your newly-created share, click Services > Control Services, then toggle the On switch next to CIFS.

To view the share on your Windows computer, go to My Computer, then type your computer's address like so: \\\$ip-address/\$mount-name. Or, you can mount the share like a drive by right-clicking "My Computer" and choosing "Map Network Drive."

To view the share on your Mac computer, open Finder. You should see the share show up in

the left-hand side of your finder. If not, go to the menu and click Go > Connect to Server. Type `smb://$ip-address/$mount-name` then click OK.

## Other Services

Here is a quick rundown of other services you might find useful on your FreeNAS implementation.

- **AFP:** The proprietary system used to share files to Mac and iOS systems.
- **Active Directory:** Allows you to use your FreeNAS server as an AD server for connected Windows machines.
- **Dynamic DNS:** Use this to connect your FreeNAS server to a Dynamic DNS service, which will allow an Internet-connected server without a static IP address to always use the same domain name.
- **FTP:** Use your FreeNAS server as an FTP server for the files it hosts.
- **iSCSI:** Connect your FreeNAS server to an iSCSI storage host.
- **LDAP:** Connect your FreeNAS server as an LDAP host, and allow it to manage your share's authentication.
- **Plugins:** This is an [advanced feature](#) that can allow you to use special plugins for other services like uPnP, DAAP, torrents, etc. The feature is still in beta.
- **Rsync:** Set up a FreeNAS share as a dedicated rsync folder for automating file synchronization between Linux clients.
- **SNMP:** Use your FreeNAS server as an SNMP share, for monitoring the status of other network devices.
- **S.M.A.R.T.:** Use the SMART disk reporting service on your FreeNAS volumes to email you when your disks are unhealthy or need to be checked.
- **SSH:** Allow logins to your FreeNAS server via SSH.
- **TFTP:** Establish a TFTP share with on your FreeNAS server. TFTP is a lightweight version of FTP used for minimal tasks like PXE network boots.
- **UPS:** Configure FreeNAS to work with a connected UPS power supply.

### 3.13.4 – Further Reading

- [FreeNAS Documentation](#)





## **The CitizenWeb Guides – Crash Courses**

### **4.1 Backup and Encrypt Your Data**

---

#### **4.1.1 – Backup**

Backing up with Linux is easy. The quickest and simplest way to do it is to simply move your data into a TAR archive. This can be accomplished with the following command:

```
tar cvzf archivename.tar.gz FILES
```

The "cvzf" is actually a list of options, which means: (c)reate an archive, print out the list of files we want to compress (v)erbosely, (z)ip up the archive with gzip, and specify our own (f)ilename.

If we want to back up a directory of files, we can easily do that with:

```
tar cvzf archivename.tar.gz --directory=/path/to/folder/ .
```

This will copy the chosen folder's contents to a new archive in the current folder. You can easily use this to backup individual folders into archives, then move them to a different drive or an offsite location.

For paranoid cases, you can choose to backup your entire system with:

```
tar cvzf bak-system.tar.gz --directory=/ .
```

Restoring from a backup is simple:

```
tar xvzf archivename.tar.gz -C /path/to/extract/dir
```

Make sure that you set the proper location for whatever data you want to restore.

### 4.1.2 – Encrypt Your Backups

It is advisable that you encrypt your backups before they leave your computer. This is especially true if you wish to use a public backup storage service like tarbackup.

You will need to decide upon a method for encryption. You can opt to enter a password each time you wish to backup or restore an archive, or you can choose to keep a password stored in a file on your harddrive. This isn't the advisable option, as it is less secure: anyone who can get their hands on your computer can potentially find and decrypt your backups if they are stored elsewhere. However, it can be helpful for automating your backups via a bash script (more on that in a future guide).

To encrypt a backup, first create the TAR archive as described above. Then use openssl to create your backup. If you are using a password:

```
openssl enc -aes-256-cbc -salt -in archivename.tar.gz -out  
archivename.tar.gz.enc -pass pass:PASSWORD
```

If you are using a password stored in a keyfile:

```
echo PASSWORD > enc.key  
openssl enc -aes-256-cbc -salt -in archivename.tar.gz -out  
archivename.tar.gz.enc -pass file:enc.key
```

Again, if you are to keep this file around, take precautions to secure it!

After the command completes, your archive file will be encrypted. Remove the .tar.gz file and store the .tar.gz.enc file as you need to.

To decrypt the archive, run:

```
openssl enc -d -aes-256-cbc -in archivename.tar.gz.enc -out  
archivename.tar.gz -pass pass:PASSWORD
```

If you are decrypting with a file rather than with an entered password, substitute "file:FILENAME" for "pass:PASSWORD".

### **4.1.3 – Options for Storing Backups**

Backing up to TAR files is convenient because you have many different storage options. After the TAR archive is created and/or encrypted, you can simply move it to whatever storage media you want. This can be an external hard drive, a DVD, another server or NAS drive, etc.

There are also a few different options for storing tar-based backups online these days. [Tarbackup.com](http://Tarbackup.com) and [Tarsnap.com](http://Tarsnap.com) are two such services. You can pay a low fee to store your encrypted backups on their high-capacity servers, ready for download at a moment's notice.