**This is a collection of guides that I used, with a few alterations and changes for my specific needs, in order to set up my encrypted amazon cloud drive, and ultimate headless Ubuntu plex server.**

**In all the commands that are listed you can substitute the user “plex” for your specific user, be sure to also replace “plex” in all of the paths that are listed.**

**Some of the guides are no longer using the most recent version of programs, check the developer websites for most recent version. And most screen shots should still expain, but slight menu options may have changed.**

**How To Setup A Headless Media Server Using Ubuntu 14.04 and Plex Software**

This is a step-by-step guide that will teach you how to setup an Ubuntu-based media server. I’m gonna use a proprietary software called Plex Media Server. It’s is a software that you can use to organize and stream your media collection. It let’s you stream media to various multimedia devices like, Smart-TV, mobile phones, streaming players, and other multimedia applications (like XBMC).

It’s an awesome piece of software that you could build awesome projects upon it. In my case, I use it with my XBMC box, Android Phone and Smart-TV. You could install Plex in various supported platforms, and you could either install it with a GUI or GUI-less (headless) installation. But in this guide, I’m going to show you how to configure via command-line. I’ve chosen Ubuntu 14.04 LTS Server edition as my base-OS for this guide. If you wanted to use other OS as your Base, then you should read additional information below to know the various platforms it currently supports. If you wanted to use a Linux-based setup, and you want it headless, then read further and follow the steps enumerated below. Also, prepare yourself to be acquainted with Linux command-line.

**Minimum System Requirements**

You also must know the hardware requirements so you could tell whether you choice of OS and hardware components have met Plex’s minimum requirements. Your choice of hardware must at least meet the minimum, to install Plex successfully. If you have higher requirements and probably wanna build a kick-ass media server, then you should go more than the minimum. Read further below to see the nominal requirements.

**Standalone Implementation – No Transcoding**

The nominal requirements would be enough to install and run Plex, if you don’t intend to encode and transcode media files with your machine. Also, if you don’t intend to install and run other CPU intensive applications and background services, the minimum requirements would be sufficient to start with. But in my opinion, I think it’s better to start with a more powerful CPU, with a minimum amount of memory. Because it is easier to upgrade a memory than a CPU. You just throw another stick of ram and you have an upgrade. This is not the case of CPU, you have to change the whole processor and buy another one to have an upgrade.

Note: These are just guidelines, only to give you an idea or ballpark estimate of what hardware specs your box may require. This serves as base unit, so it’s better to use more than the minimum, so long as you could and has the resources to acquire the hardware components. **Requirements:** \* At least 1.6Ghz 2-Core processor (Intel Core 2 Duo or AMD X2 Series) \* At least 1 GigaByte of Memory, if you’re on Mac or Windows \* At least 512MegaByte of Memory, for Linux \* Hard Drive, at least 80GB

**With HD Encoding or Transoding Capability**

If you wanted your Plex box to have the HD encoding or transcoding capability and at the same time, serves and stream media to several media clients, the following requirements are suggested.

**Requirements:**

* For single instance 720p HD file transcoding: 2-Core 2.0GHz CPU
* For single instance 1080p HD file transcoding: 2-Core 2.4Ghz CPU
* For multi-instance 720p & 1080 HD file transcoding: At least 4-Core CPU
* At least 2GigaByte of Memory
* H.264 Capable Video Card, if you want a GPU based encoding. (More about this in my next post.)

**Plex Media Server, Headless Ubuntu 14.04**

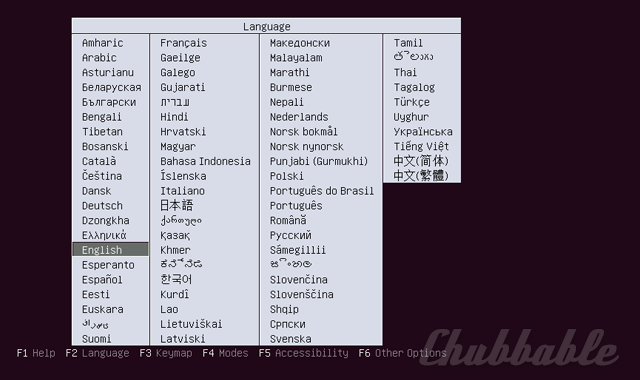
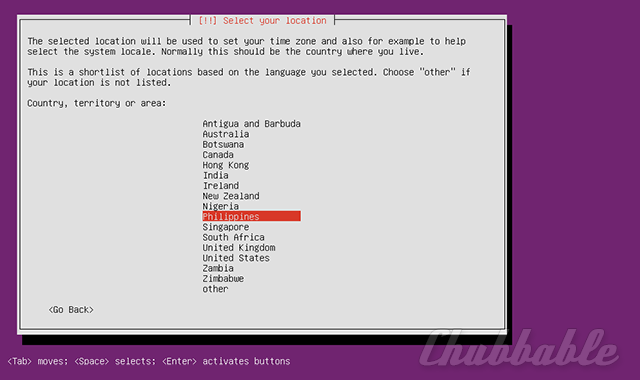
Installation If you have the box that met the minimum requirements, then you may now proceed with the following procedures. Plex Media Server supports Windows, Mac, Linux and FreeBSD, but in this guide, I’m going to show how you would install it in a server edition of Ubuntu 14.04 LTS. The following steps below will walk you through, on how to set things up. This is going to be a headless and standalone setup. That means, no monitor and peripherals attached. But we need to attach those during the installation and initial setup. Before you start with this guide, you must have already downloaded the latest version of Ubuntu (14.04.1 LTS as of July 2014) and burned it to a DVD disc, and your machine meets the nominal system requirements for a stand-alone PMS installation. Please read the necessary things you need for this guide, below.

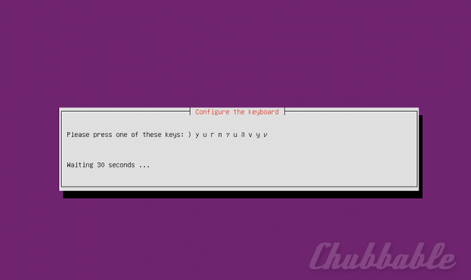
**Things you need:**

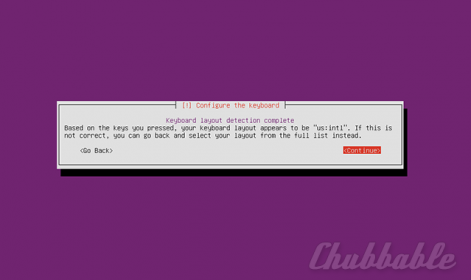
* Comfortable wit Linux commands
* Putty (from http://putty.org), if you’re on Windows
* Extra device for testing (Android or iOS device, streaming player or similar)
* A clean and fresh hard-drive with at least 80GB free space
* Separate hard drive for media files (optional)
* Hardware components that meets the minimum requirements
* A configured DHCP server This guide is broken into a two-part series. For the first part, we will install our operating system, and that would be Ubuntu server edition. The second part would be the installation of the Plex software.

**Part 1: Ubuntu Server 14.04 LTS Installation**

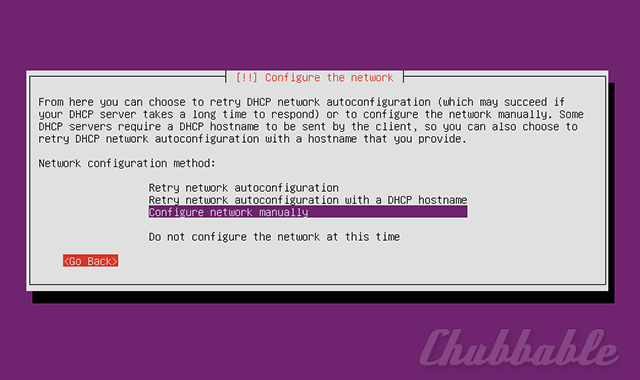
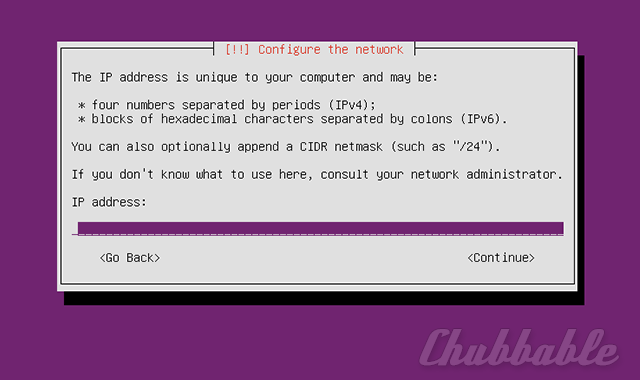
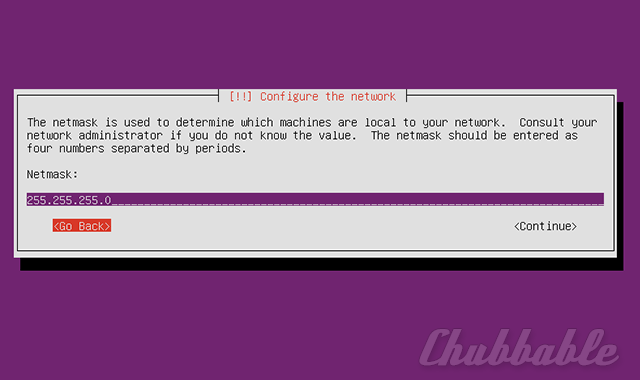
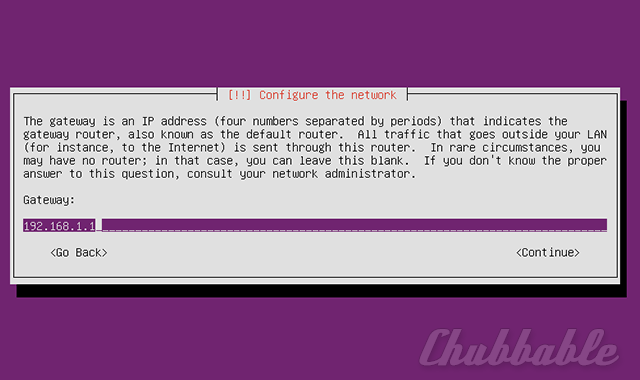
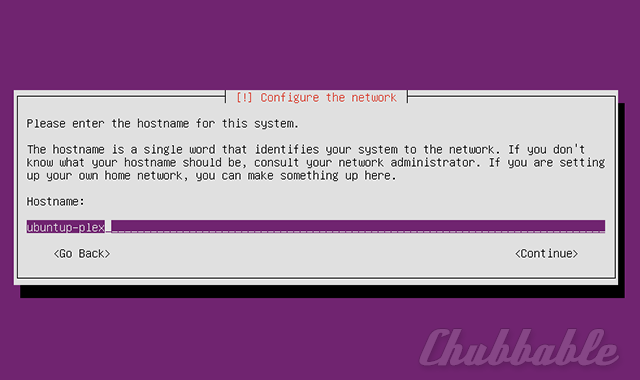
For the first part, we install Ubuntu 14.04 as our base-OS. Below are the necessary procedures.

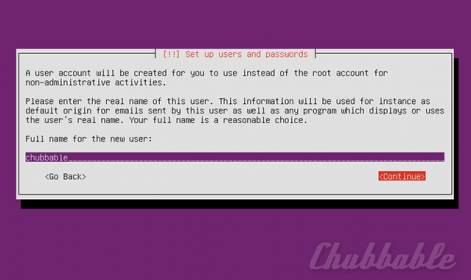
* **Step 1:** Power on your machine and set your BIOS settings to boot from CD/DVD.
* **Step 2:** After doing so, insert your Ubuntu DVD installer and boot from it. When your machine successfully booted up, you’ll be prompted with a menu, similar to the image shown below.Select your desired language using your keyboard’s arrow keys. Navigate and highlight your language and then press enter to activate it. In my case, I choose English.
* **Step 3:** Remember that we’re setting a headless setup of Ubuntu, so we’ll setup just the core files, plus OpenSSH server. So just choose Install Ubuntu Serverby highlighting it with your arrow keys, and press enter.That should take you to the next step.
* **Step 4.** At this point, you must choose your location. It is recommended to choose your real geographic location because Ubuntu will choose the most optimal mirror server for your current location. So if you choose United States, Ubuntu shall configure the apt sources list to pick the most optimal servers that are located in the U.S.So in my case I’d choose Philippines.
* **Step 5.** The next prompt will ask you if you want your keyboard be detected properly by the installer. If you have chosen a different language other than English from the previous steps, and you have a different keyboard layout other than the QWERTY layout, I suggest that you choose Yes here. You should use TAB key to select a choice from this menu. If you select YES, then you will be ask to press a series of keys. The installer will detect your keyboard layout properly depending on what keys you’ve pressed. In my case, I would choose No.

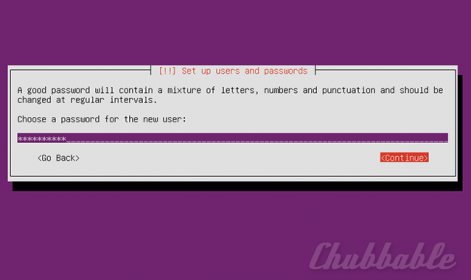
[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/05-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/05-plex-media-server-headless-ubuntu-install.png" \o "Ubuntu Install Keyboard Detech)

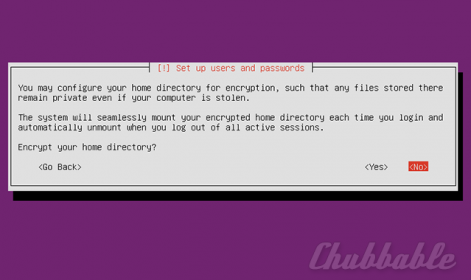
[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/06-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/06-plex-media-server-headless-ubuntu-install.png" \o "Ubuntu Install Keyboard Detech)

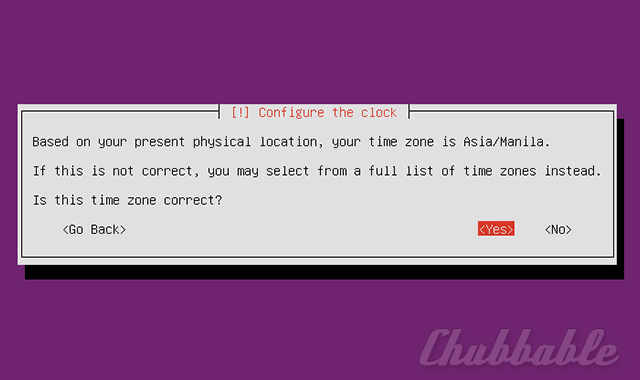
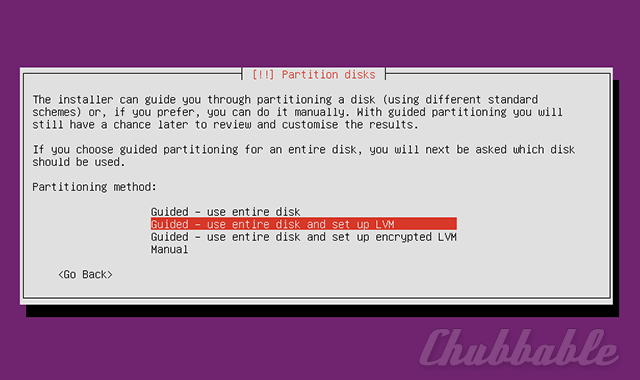
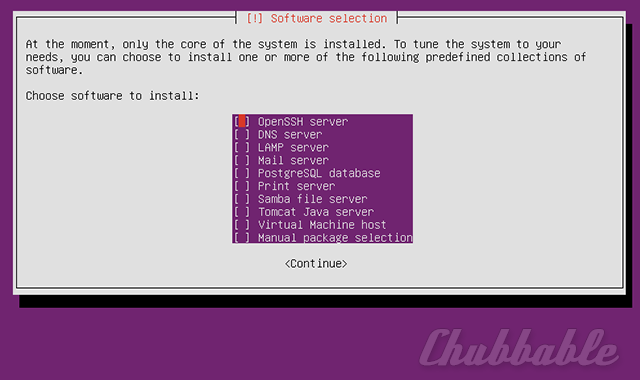
Now, depending on your answer, if you choose NO (like I did) the installer will ask you again if what is the Country Origin of your keyboard, In my case, I should choose English US and hit enter and the installation should start. But if you choose YES to let the installer detect your keyboard, it will not ask you to select your keyboard’s Country of origin, instead it will start the installation process.

* **Step 6.** After the above step, depending again on your setup, the installer will ask you to configure your network manually if it didn’t acquire a dynamic IP address. But if you have configured a DHCP server inside your network, and the installer automatically acquires an IP address, no prompt will pop-up.In case you need to configure your network manually, then you should follow the following steps.
  + **6.1:** Select and activate Configure network manually.
  + **6.2:** Then the installer will prompt you to input your desired IP address. Enter your IP address, it should have the same subnet as your existing network. In my case, I have a subnet of 192.168.1.xxx, so I would input my IP like so, 192.168.1.99. Then hit enter to continue with the next step. Use TAB key to select CONTINUE.
  + **Step 6.3:** Enter a subnet mask, in my case, 255.255.255.0.
  + **Step 6.4:** Enter your gateway IP address, this should be your router’s IP address. In my case, 192.168.1.1.
  + **Step 6.5:** Enter you name server address, this should also be your router’s IP address. In my case, 192.168.1.1.
  + **Step 6.6:** Select and activate Continue button. You should be taken to the next step. You’re done with the manual network configuration.
* **Step 7:** You’ll be prompted to enter your desired host name with this step, so enter your host name. This will be the name of your box/machine that you could identify from your local network. You may name it like, ubuntu-headless or any name you can think of. It could be planet names or celebrities. In my case, I’ll name it ubuntu-plex.
* **Step 8:** The next step will ask you for your domain name. But since this host will only be used inside the local network, we could just make something up. So in my case, I would name it chubbable.loc. Now we have configured our domain name. And when a domain name is attached to a host name, it makes the host name a [fully qualified domain name](http://en.wikipedia.org/wiki/Hostname). But this setup is just for local network. You don’t really have to register a domain name. In my case, my fully qualified domain name for my setup would be ubuntu-plex.chubbable.loc.
* **Step 9:** Next, enter your name, your desired username and password.

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/14-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/14-plex-media-server-headless-ubuntu-install.png" \o "14-plex-media-server-headless-ubuntu-install)

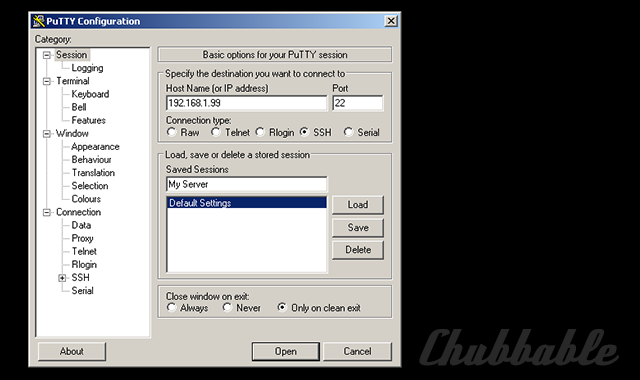
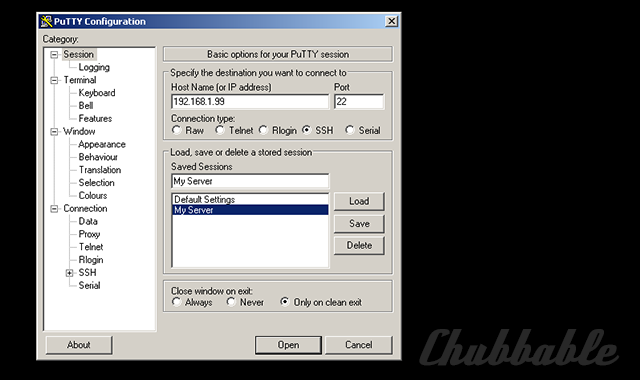
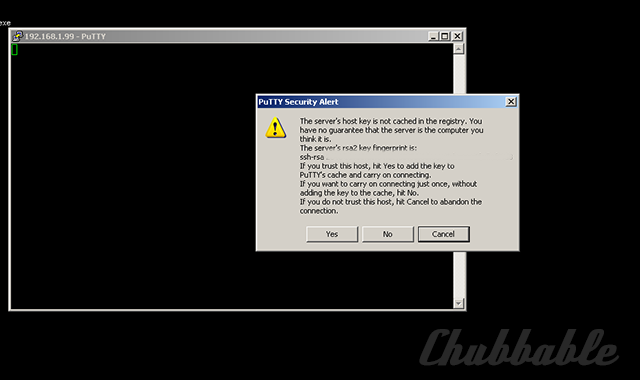
[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/16-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/16-plex-media-server-headless-ubuntu-install.png" \o "16-plex-media-server-headless-ubuntu-install)

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/17-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/17-plex-media-server-headless-ubuntu-install.png" \o "17-plex-media-server-headless-ubuntu-install)

* **Step 10:** The installer will ask you if you want your home directory encrypted, since this is just a basic and simple setup, we may choose No to not encrypt home directory.
* **Step 11:** Choose your time zone. If you want to choose another time zone, then you should choose NO when asked if Is this time zone correct?In my case, it automatically selects my desired time zone, which is Asia/Manila, so I’d answer Yes.
  + Note: The installer can only select your Country automatically when your machine is connected to the Internet. It selects your Country based on your detected public IP address.
* **Step 12:** Choose a partitioning method. By default, Guided - use entire disk and setup LVM option is selected. I recommend to just implement it and hit enter to start partitioning. Select your hard disk and press enter to continue. In my case, I have a 80GB Sata hard disk. When asked to Write changes to disk and configure LVM, answer Yes.Another prompt will ask you, Write the changes to disk?, answer Yes and the partitioning process should start. After this process, a prompt will ask you to configure a HTTP proxy, you may skip this by hitting Continue button and proceed with the installation.
* **Step 13:** After the previous step the install process should start, you can sit back and relax while the installer applies your settings. During this process, another prompt will pop-up asking you how you would manage upgrades on your system. You should choose Install security updates automatically. Another prompt will ask you if what software to install. Since this is going to be an Ubuntu headless setup, you would select OpenSSH Server package, and you may select Samba file server as well (you can use this service later on with your media file management). To select a package, use arrow keys and space-bar to activate it and hit Continue button to continue.In the final part of this process, another prompt will ask you if you wanted to install GRUB bootloader to the master boot record. You should answer Yes to continue. That would be it, hit Continue once more to finalize the install process. Your machine should reboot and should take you to a black-screen login prompt. Congratulations! You have now installed a GUI-less Ubuntu 14.04. You may now continue with the next phase of the server setup.

**Part 2 – A: Plex Installation Procedures for Windows Platform**

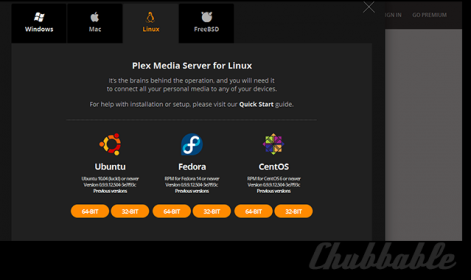
If you’re on windows, you need to have Putty.exe to be able to remotely connect to your headless server.

* **Step 1:** Download and Install putty.exe. You can download the software from http://www.putty.org. You dont’ need to install it, just double-click the putty.exe to launch it and follow Step 2 below.
* **Step 2:** Enter your Ubuntu server’s IP address. In my case it is 192.168.1.99. Fill up Save Sessions and enter something like Ubuntu Headless or My Serveror any text you might want, then click save.
* **Step 3:** A new entry should appear right below the entry, Default Settings. Look for the entry that you’ve created from Step 2 above. Select and click Loadbutton, then click Open button. A new window and prompt should appear. Click Yes to continue.
* **Step 4:** You should be now at the log-in prompt. Enter your username and password. The one you entered during the Ubuntu 14.04 server installation. When you’re logged in, you should see a window similar to the image shown below.

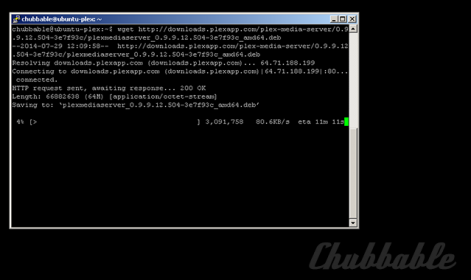
**Install Plex Software Using SSH In Windows** – Do the steps below. Let’s download the latest Debian package (.deb file) for our OS (Operating System).

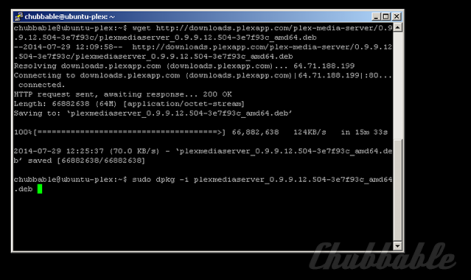
* **Step 1:** Fire-up your browser and head over to plex.tv/downloads. Then click COMPUTER button, a pop-up window should open.
* **Step 2:** Click Linux tab and scroll further below, right-click on 64-BIT button just below the Ubuntu text and select Copy link location. Minimize the window and switch-back to your Putty session.

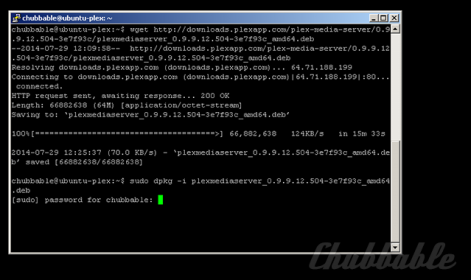
[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/35-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/35-plex-media-server-headless-ubuntu-install.png" \o "Download Link)

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/36-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/36-plex-media-server-headless-ubuntu-install.png" \o "Ubuntu 64-bit Download Link)

* **Step 3:** Type-in wget and then space. Then paste what you have just copied (copy link location) in the Putty.exe window. To do this, just right-click on it to paste, then hit enter. It should download the latest version of Plex software. Wait for it to finish then proceed to the next step.

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/38-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/38-plex-media-server-headless-ubuntu-install.png" \o "Plex Download and Install Process - Image 2)

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/39-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/39-plex-media-server-headless-ubuntu-install.png" \o "Plex Download and Install Process - Image 3)

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/40-plex-media-server-headless-ubuntu-install.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/40-plex-media-server-headless-ubuntu-install.png" \o "Plex Download and Install Process - Image 4)

* **Step 4:** After downloading the file, install it by using the following command: sudo dpkg -i plexmediaserver\_0.9.9.12.504-3e7f93c\_amd64.deb and hit enter. You’ll be prompted to enter your password, enter your password and hit enter.

Note: When you type your secret-pass, your screen won’t print it.

Installation should start when you typed-in your pass-phrase correctly.

TIP: Noticed that the Plex software installer has a very long file name, to help you out a little bit, you may do the following: Type sudo dpkg -i then space, then type plex, and then hit Tab key. It should auto-completes the file-name.

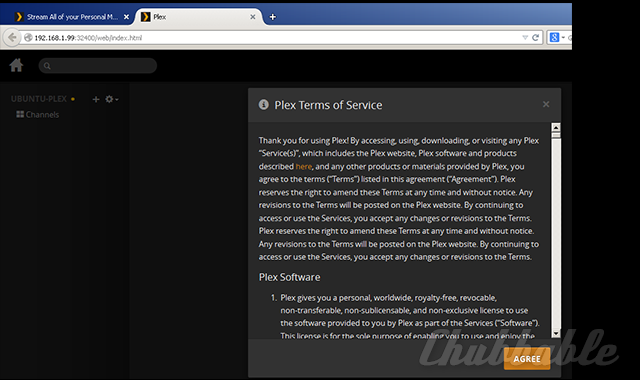
* **Step 5:** If you encountered an error during install process, do this: type sudo apt-get install -f and hit enter. When ask if you want to continue, answer yand hit enter. Installation should resume.
* **Step 6:** After the above installation step, we now test the server and see if Plex was installed properly. So let’s do that now and fire-up a browser. To access Plex’s web-based control panel, you would type-in your server’s IP address plus port number, then directory name manage The format is like this:
* ip-address:port-number/manage

So you would type like this in your browser’s address bar:

192.168.1.99:32400/manage

Substitute 192.168.1.99 with your machine’s IP address. Port Number 32400 is Plex’s default port it listens to, for web administration.

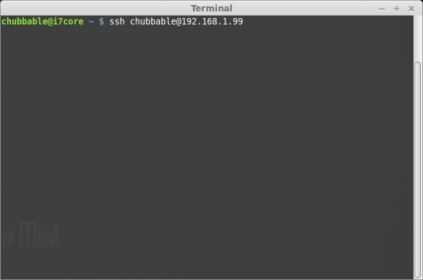
If you see a web-page similar to the image below, then you have setup Plex media server correctly. Now click

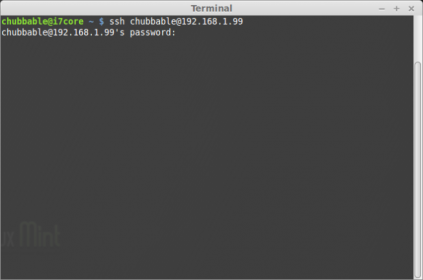
Agree button to accept the software terms of service.At this point, you can now configure and administer your server remotely from any machine within your local network. You’ll be using a lot of command-line syntaxes when you configure your Linux server and when you add or install additional modules/packages to it.

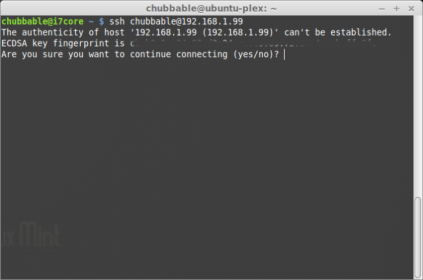
**Part 2 – B: Installation procedures for Ubuntu-based Platform:**

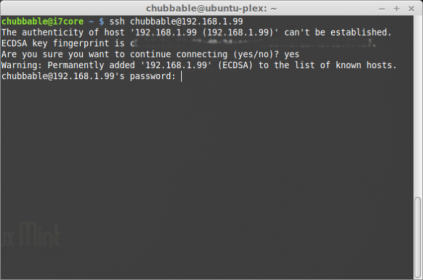
Installing Plex Media Server from an Ubuntu-based OS (Ubuntu Deskop or Linux Mint) is very similar to the steps we’ve taken previously. The only difference is, we don’t need to download

Putty.exe or similar software. We only need to open a terminal and type in… ssh user-name@192.168.1.99. A window will pop-up asking you if you wanted to continue, just type yes and hit enter. You could now move on and follow the procedures from “**To Install Plex Using SSH In Windows**” above. Sample Screen From LinuxMint Command Prompt:

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/linuxmint-command-commandline-ssh-653x432.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/linuxmint-command-commandline-ssh-653x432.png" \o "linuxmint-command-commandline-ssh)

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/linuxmint-command-commandline-ssh-password-653x432.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/linuxmint-command-commandline-ssh-password-653x432.png" \o "linuxmint-command-commandline-ssh-password)

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/linuxmint-command-commandline-ssh-fingerprint-653x432.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/linuxmint-command-commandline-ssh-fingerprint-653x432.png" \o "linuxmint-command-commandline-ssh-fingerprint)

[[](https://chubbable.com/chubbable-blog-content/uploads/2014/08/linuxmint-command-commandline-ssh-accept-connection-653x432.png)](https://chubbable.com/chubbable-blog-content/uploads/2014/08/linuxmint-command-commandline-ssh-accept-connection-653x432.png" \o "linuxmint-command-commandline-ssh-accept-connection)

Tip: You could copy-paste an applicable command to a command-line. To paste a copied text or snippet to a Linux terminal, click middle mouse button.

I have my media files on a separate windows server that I am using as a NAS. So I have to use the following steps to read those files from the plex server.

# Introduction

This guide will show you how to setup a mount of a remote windows share, and have it always there when you startup.

## Two methods, depending on share host

1. cifs
2. smbfs

smbfs is the "original" method.

However, smbfs is not compatible with security signatures, which are enabled by default and not recommended to disable on Windows Server 2003 and later. If a share is served by Windows Server 2003 or later, you should use cifs.

# Prerequisites

You must have a windows machine (or other machine running Samba) with an accessible share.

The 'samba' package itself is not necessary if you only need a smb client.

The package providing the tools needed to mount "smbfs" and "cifs" filesytems is "smbfs" (up to 10.04) or "cifs-utils" (10.10 onwards). You may have smbfs installed on your machine. If not, run

sudo apt-get install smbfs

...or...

sudo apt-get install cifs-utils

...as appropriate.

Update the unmount order to prevent CIFS from hanging during shutdown.

sudo update-rc.d -f umountnfs.sh remove

sudo update-rc.d umountnfs.sh stop 15 0 6 .

# Setup

## Single User

Note the UID of the single user which is to have access to the share. For a user named $username, the following command outputs the UID

grep $USER /etc/passwd | cut -d: -f3

## Multiple Users

If multiple users are to have the same level of access to the share, then create a new user group, presumably named after the share.

Navigate to "System" -> "Administration" -> "Users and Groups" -> "Manage Groups". -> "Add Group" and select a name, Group ID (GID), and group members. Note the GID -- you will need it later.

## Credentials File

Warning- this method is not completely secure, any user with root access could see your password in plain text.

Create a file called .smbcredentials, probably in the home directory of the primary user of the share. In this file put username an equals sign and the windows username (and domain if loging into a domain) on the first line, put password an equals sign and the password for that user account on the second line of the file. The file should look like:

username=MyUsername

password=MyPassword

# OR:

# username=MyUsername@MyDomain

# password=MyPassword

# OR: (for cifs on Windows Serve 2003)

# username=MyDomain/MyUsername

# password=MyPassword

On the command line, in the directory of .smbcredentials type

sudo chown root .smbcredentials

sudo chmod 600 .smbcredentials

this will ensure that only root can access this file.

***Note:***Regretfully as from version 3.3.2-1ubuntu3.2 (October 2009) this approach is no longer possible together with the "user" option. A security fix prevents reading the credentials file if you don't have read access to it. You will have to pin the packages at version 3.3.2-1ubuntu3 or 3.3.2-1ubuntu3.1 to continue using this approach as non-root.

## Editing fstab

Warning- editing the fstab file can be dangerous, please back it up before continuing.

Note: if servername or sharename has a literal space (i.e. ' '), substitute \040 instead, so that 'server name' becomes 'server\040name'

Add a line at the bottom of your /etc/fstab file that specifies:

//$SERVER/$SHARE $MOUNTPOINT $FS\_TYPE credentials=$SMB\_CREDENTIALS,uid=$UID,gid=$GID

# e.g.

SERVER=apollo

SHARE=install\_files

MOUNTPOINT=/path/to/mnt

FS\_TYPE=smbfs

SMB\_CREDENTIALS=/path/to/.smbcredentials

UID=1000

GID=1000

### smbfs, group perms

1. FS\_TYPE=smbfs
2. GID=1234 # the newly created group's ID
3. don't include uid=$UID, which defaults to that of root

//apollo/install\_files /path/to/mnt smbfs iocharset=utf8,credentials=/path/to/.smbcredentials,gid=1234 0 0

Note: many directories are set so that only the user can write to the directory and that the group can only read (permissions 0755), if this is the case then when it is mounted the group will still not be able to write to the directory regardless of their permission on the share. To give the group write permissions on the mount then use the following.

//apollo/install\_files /path/to/mnt smbfs iocharset=utf8,credentials=/path/to/.smbcredentials,dir\_mode=0775,gid=1234 0 0

### smbfs, user perms

1. FS\_TYPE=smbfs
2. UID=1000 # particular user's uid
3. don't include gid=$GID, which defaults to $UID

//apollo/install\_files /path/to/mnt smbfs iocharset=utf8,credentials=/path/to/.smbcredentials,uid=1000 0 0

### cifs, group perms

1. FS\_TYPE=cifs
2. GID=1234 # the newly created group's ID
3. don't include uid=$UID

//apollo/install\_files /path/to/mnt cifs iocharset=utf8,credentials=/path/to/.smbcredentials,gid=1234 0 0

Note: many directories are set so that only the user can write to the directory and that the group can only read (permissions 0755), if this is the case then when it is mounted the group will still not be able to write to the directory regardless of their permission on the share. To give the group write permissions on the mount then use the following.

//apollo/install\_files /path/to/mnt cifs iocharset=utf8,credentials=/path/to/.smbcredentials,dir\_mode=0775,gid=1234 0 0

### cifs, user perms

1. FS\_TYPE=cifs
2. UID=1000 # the user's uid
3. don't include gid=$GID

//apollo/install\_files /path/to/mnt cifs iocharset=utf8,credentials=/path/to/.smbcredentials,uid=1000 0 0

## Ensure

1. The entire expression MUST all be on one line in your fstab file
2. use "//" and "/" instead of "\\" and "\" when specifying the share location
3. /path/to/mnt is a directory that exists (and is empty)

## Completing Setup

Reload fstab:

sudo mount -a

# Troubleshooting

## cifs will not mount

Note:- cifs by default does not resolve netbios names so you may get an error message when you try to mount that the name could not be resolved into an address and "could not find target server". In order to enable netbios resolution you need to edit /etc/nsswitch.conf and add the winbind package:

1. edit /etc/nsswitch.conf

sudo gedit /etc/nsswitch.conf

change the line from

hosts: files dns

to

hosts: files wins dns

1. next install winbind

sudo aptitude install winbind

Now you should be able to mount the directory.

***Note:***If you experience slow dns resolution after making these changes, you can change the order of the entries to the following and you may see an improvement.

hosts: files dns wins

## Server is down, filesystem is hung

If the client somehow loses contact with the Samba server, then the filesystem will probably get hung. Basically, it becomes a blackhole, eating things that try to read to/write from it (e.g. ls) and refusing to go away (e.g., umount says that the "device is busy").

Sometimes, all you need to do is restart the Samba daemon on the server machine.

sudo /etc/init.d/samba restart

If that doesn't work, or for some reason you can't do anything on the server side, then try

sudo umount -lf /mount/point

The -f option forces (possibly unclean) unmounting, and the -l option is for "lazy unmounting", and seems to work around "device is busy" errors that occur with just -f.

## CIFS remote ownership enforcement

When you connect using CIFS to a server which supports Unix permissions (e.g. Samba), CIFS will by default try to enforce remote Unix ownership UIDs and Unix permissions when you try to access the share. i.e. if a file is owned by UID 502 on the remote server, then the local kernel will try to enforce the same permissions if it were owned by UID 502 on the local machine. Note: This has nothing to do with the remote server's security settings. This is an extra local ownership enforcement by the filesystem driver. It is a feature to allow use of remote share as a local drive with full Unix permissions enforcement if users match.

But if this is a public share, then chances are, the remote UIDs will not make sense locally. A remote UID might be a completely different user or might not exist at all on the local machine. If remote UIDs and local UIDs do not match, then local users will have trouble using the share. To disable this, use the "noperm" mount option. Remote permissions and UIDs will still be visible, but they will not be enforced locally.

## System Hangs on Shutdown

Sometimes during shutdown, networking will be turned off before the network share is unmounted. This will cause the computer to display the below code for a few minuets before shutting down (the numbers seem to change after each boot).

CIFS VFS: server not responding

CIFS VFS: no response for cmd ## mid ###

To fix this problem, and allow the computer to shut down smoothly, just change when the network share is unmounted by the file system. This can be done by running the following commands:

sudo update-rc.d -f umountnfs.sh remove

sudo update-rc.d umountnfs.sh stop 15 0 6 .

A better solution for those using Gnome: <http://ubuntuforums.org/showthread.php?t=1347340>

## Login without Credentials

If you want to mount the share without the credentials file you can use the entry below. I believe that by adding the **\_netdev** in the entry below, it will not mount the share if you are not connected to the same network that the share is on or if you are not connected to a network at all.

1. # /etc/fstab: static file system information. #

# <file system> <mount point> <type> <options> <dump> <pass>

//<server>/<share> <mount point> cifs rw,\_netdev,user=<username>,password=<password>,uid=<uid>,gid=<gid> 0 0

Here is an example of the last line //gurnee/projects /home/jcrow/GurneeServer cifs rw,\_netdev,user=DOMAIN/user,password=password,uid=1000,gid=100 0 0

The server being connected to is Gurnee, the shared folder is projects, the mount point is /home/jcrow/GurneeServer

## Connect when network available

The **\_netdev** option is also used for systems that only have networking started at user login (as when using the Gnome Network Manager package). For having network connections enabled at boot up (without requiring a user login) then tools that write to the /etc/network/interfaces file may have to be used. It is probably good policy to always use **\_netdev** for all automatic network mounts.

This is the contents of my fstab. Feel free to copy the contents and edit for your specific build, also remember to add make the directories you are mounting to first before you edit fstab.

# /etc/fstab: static file system information.

#

# Use 'blkid' to print the universally unique identifier for a

# device; this may be used with UUID= as a more robust way to name devices

# that works even if disks are added and removed. See fstab(5).

#

# <file system> <mount point> <type> <options> <dump> <pass>

# / was on /dev/sda1 during installation

UUID=c3c6333d-2cd1-4789-a00d-42ebd92bbfb9 / ext4 errors=remount-ro 0 1

# swap was on /dev/sda5 during installation

UUID=8b10f7dc-efd2-4984-8275-454fbfde2e24 none swap sw 0 0

//PLEXMEDIA/Cartoons /home/acd/backup/PLEX/Cartoons cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

//PLEXMEDIA/Movies /home/acd/backup/PLEX/Movies cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

//PLEXMEDIA/TVShows /home/acd/backup/PLEX/TVShows cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

//PLEXMEDIA/Workout /home/acd/backup/PLEX/Workout cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

//PLEXMEDIA/Music /home/acd/backup/PLEX/Music cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

//PLEXMEDIA/workoutdocs /home/acd/backup/PLEX/Music cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

//PLEXMEDIA/Pictures2 /home/acd/backup/Pictures cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

//PLEXMEDIA/Documents2 /home/acd/backup/Documents cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

//PLEXMEDIA/GameRoms /home/acd/backup/GameRoms cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

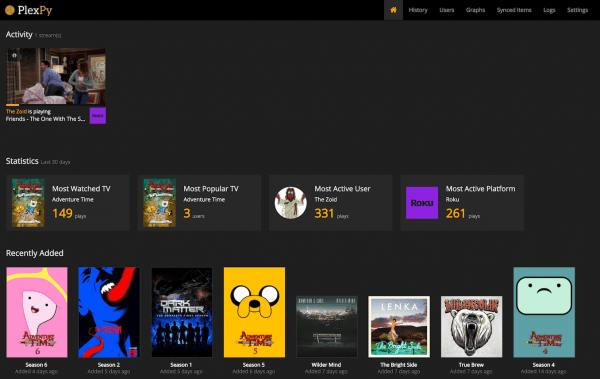
//PLEXMEDIA/TRANSFERS /home/acd/backup/Transfers cifs credentials=/home/acd/.smbcredentials,iocharset=utf8,sec=ntlm 0 0

# Install PlexPy on Ubuntu for Plex Monitoring

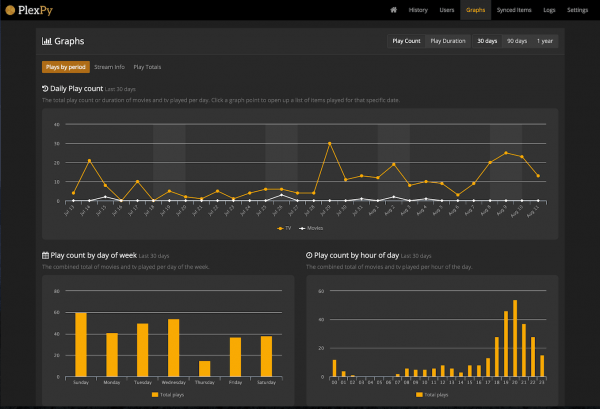


Install [PlexPy](https://forums.plex.tv/discussion/169591/plexpy-another-plex-monitoring-program) on Ubuntu for [Plex](https://plex.tv/) Monitoring to add some extra sleekness and professionalism to your home media server. PlexPy is a python-based version of [PlexWatch](https://github.com/ljunkie/plexWatch) on [github](https://github.com/drzoidberg33) and uses code from [Headphones](https://github.com/rembo10/headphones) for the web interface and daemon. You can use PlexPy to send notifications to [Kodi](https://kodi.tv/) and push services like [Pushbullet](https://www.pushbullet.com/) and [Pushover](https://pushover.net/) as well as email. This PlexPy installation tutorial was tested on Ubuntu 16.04 and should work on all previous versions of Ubuntu and even Debian-based systems.

PlexPy can show you tons of information about your Plex Media Server, here is a sample overview screen.

[](http://www.htpcguides.com/wp-content/uploads/2016/05/plexpy-overview-screenshot.png)

Here is a screenshot of the statistics page showing an overview of media play counts.

[](http://www.htpcguides.com/wp-content/uploads/2016/05/plexpy-statistics-screenshot.png)

## Install PlexPy on Ubuntu

Update repos then install git and python on Ubuntu

sudo apt-get update

sudo apt-get install python git -y

Clone the latest PlexPy from the [github repository](https://github.com/drzoidberg33/plexpy/)

sudo git clone https://github.com/drzoidberg33/plexpy.git /opt/plexpy

Create a secure PlexPy user that has no shell and isn’t allowed to log in which will be used to run the PlexPy daemon

sudo adduser --system --group --disabled-login plexpy --home /opt/plexpy --shell /bin/nologin --quiet

Change ownership of PlexPy to our secure plexpy user

sudo chown -R plexpy:plexpy /opt/plexpy

Create the default PlexPy file which the init.d script reads from to set certain options like the user it runs as, port it runs on and so on.

sudo nano /etc/default/plexpy

Paste the PlexPy default configuration file below.

If you are running Headphones then port 8181 is likely occupied so change HP\_PORT to something other than 8181

HP\_USER=plexpy #$RUN\_AS, username to run plexpy under, the default is plexpy

# HP\_HOME= #$APP\_PATH, the location of PlexPy.py, the default is /opt/plexpy

# HP\_DATA= #$DATA\_DIR, the location of plexpy.db, cache, logs, the default is /opt/plexpy

# HP\_PIDFILE= #$PID\_FILE, the location of plexpy.pid, the default is /var/run/plexpy/plexpy.pid

# PYTHON\_BIN= #$DAEMON, the location of the python binary, the default is /usr/bin/python

# HP\_OPTS= #$EXTRA\_DAEMON\_OPTS, extra cli option for plexpy, i.e. " --config=/home/plexpy/config.ini"

# SSD\_OPTS= #$EXTRA\_SSD\_OPTS, extra start-stop-daemon option like " --group=users"

HP\_PORT=8181 #$PORT\_OPTS, default is 8181 for the webserver, overrides value in config.ini

Ctrl+X, Y and Enter to Save

Copy the Ubuntu PlexPy init.d script over

sudo cp /opt/plexpy/init-scripts/init.ubuntu /etc/init.d/plexpy

Make the Ubuntu PlexPy init.d script executable

sudo chmod +x /etc/init.d/plexpy

Now enable the PlexPy init.d script on Ubuntu

sudo update-rc.d plexpy defaults

Finally start the PlexPy service on Ubuntu

sudo service plexpy start

PlexPy will be running on boot, if you find that it isn’t you can change the priority for its init.d script

sudo update-rc.d plexpy defaults 98

PlexPy will be running on its default port 8181 unless you changed the HP\_PORT value in /etc/default/plexpy

# Presenting AtoMiC ToolKit 2.0 – with GUI and more features

MODIFIED ON FEBRUARY 7, 2016 BY [ANAND](http://www.htpcbeginner.com/author/anand/) [15 COMMENTS](http://www.htpcbeginner.com/atomic-toolkit-2-0-with-gui/#disqus_thread)

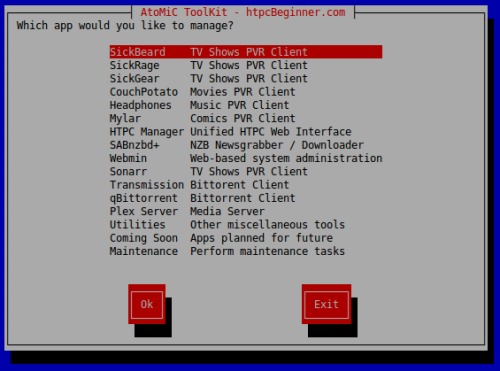


Published On: February 5, 2016

* **Share:**

* hare htpcBeginner

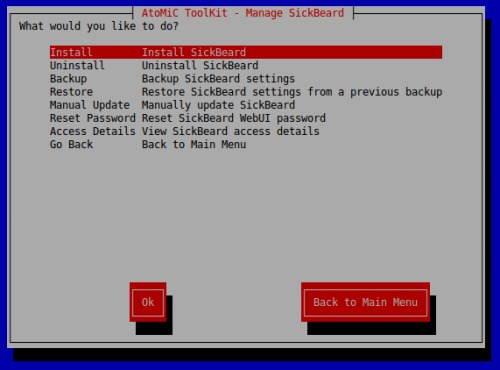
After editing over 200 files, 3000 lines of code, and 150 hours of hard work I am happy to announce the release of AtoMiC ToolKit 2.0. Many have praised the [AtoMiC ToolKit](http://www.htpcbeginner.com/atomic-toolkit/) for how easy it makes managing apps such as CouchPotato, SickBeard, Torrent clients, and more. After the initial release in April 2015, it was updated in [October 2015](http://www.htpcbeginner.com/atomic-toolkit-headphones-mylar/) to add Headphones and Mylar support and [January 2016](http://www.htpcbeginner.com/atomic-toolkit-january-2016/)with a completely rewritten code and Webmin support. A Now, the bigger and better AtoMiC ToolKit 2.0 now sports a GUI (sort of) menu that is easy to navigate and choose actions you want to perform. Take a look for yourself in the screenshots below.

[](http://cdn.htpcbeginner.com/images/2015/04/atomic-toolkit-main-menu.png)

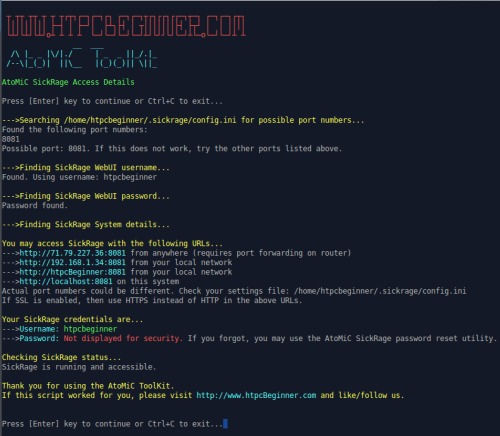
**AtoMiC ToolKit Main Menu**

## AtoMiC ToolKit 2.0 Features

* Works on Ubuntu, Ubuntu Server, and other Ubuntu variants including Xubuntu, Lubuntu, Kubuntu, and more
* Almost entirely automated
* Very little commandline work
* User- and newbie-friendly
* One-stop shop for installing and managing home server and HTPC apps
* Reset forgotten WebUI passwords
* Makes moving apps between different computers easier
* Correctly sets required permissions for trouble-free operation
* And best of all its FREE! (please consider donating if toolkit helps you.

[](http://cdn.htpcbeginner.com/images/2015/04/atomic-toolkit-sickbeard-sub-menu.png)

**AtoMiC ToolKit Submenu Example**

[](http://cdn.htpcbeginner.com/images/2015/04/atomic-toolkit-sickrage-access-details.png)

**AtoMiC ToolKit Script Execution Example**

The following apps are now fully supported: Install, Uninstall, Backup, Restore, Reset Password, Manually Update and View Access Details.

* Sick Beard – Install, Uninstall, Reset Password, Update, and Backup / Restore
* SickRage – Install, Uninstall, Reset Password, Update, and Backup / Restore
* SickGear – Install, Uninstall, Reset Password, Update, and Backup / Restore
* Sonarr (formerly NzbDrone) – Install, Uninstall, Update, and Backup / Restore
* CouchPotato – Install, Uninstall, Reset Password, Update, and Backup / Restore
* Transmission WebUI – Install, Uninstall, Reset Password, Update, and Backup / Restore
* qBittorrent WebUI – Install, Uninstall, Reset Password, Update, and Backup / Restore
* SABnzbd+ – Install, Uninstall, Reset Password, Update, and Backup / Restore
* Headphones – Install, Uninstall, Reset Password, Update, and Backup / Restore
* HTPC Manager – Install, Uninstall, Update, and Backup / Restore
* Mylar – Install, Uninstall, Reset Password, Update, and Backup / Restore
* Webmin – Install, Uninstall, Update, and Backup / Restore

My hope is to add support for the following: Planned – Plex, Deluge, ShellInABox, Emby, Monit, Apache, MySQL, nzbHydra, MusicBrainz, NZBGet, RTorrent/RUTorrent, etc. in future. Ultimately, I would like the ToolKit to allow automatically setting up the whole ecosystem for you.

### Complete Guide to Setting Up a Home Server on VirtualBox Virtual Machine

1. Install VirtualBox - [Windows 7/8/10](http://www.htpcbeginner.com/install-virtualbox-on-windows/) / [Ubuntu/Debian](http://www.htpcbeginner.com/install-virtualbox-on-ubuntu-debian/)
2. Install VirtualBox Extension Pack - [Linux and Windows Host](http://www.htpcbeginner.com/install-virtualbox-extension-pack-on-linux-windows/)
3. Install Guest OS on Virtual Box - [Ubuntu Server](http://www.htpcbeginner.com/install-ubuntu-server-on-virtualbox/)
4. Install VirtualBox Guest Additions - [Windows Guest](http://www.htpcbeginner.com/install-virtualbox-guest-additions-on-windows/) / [Linux Guest](http://www.htpcbeginner.com/install-virtualbox-guest-additions-on-ubuntu-debian/)
5. Update VirtualBox Guest Additions - [Windows Guest](http://www.htpcbeginner.com/update-virtualbox-guest-additions-on-windows/) / [Linux Guest](http://www.htpcbeginner.com/update-virtualbox-guest-additions-on-ubuntu-linux/)
6. Setup VirtualBox Shared Folder - [Windows and Linux](http://www.htpcbeginner.com/setup-virtualbox-shared-folders-linux-windows/)
7. Mount VirtualBox Shared Folder - [Windows Guest](http://www.htpcbeginner.com/mount-virtualbox-shared-folder-on-windows/) / [Ubuntu Guest](http://www.htpcbeginner.com/mount-virtualbox-shared-folder-on-ubuntu-linux/)
8. Access USB Drive on VM - [Windows and Linux Guest](http://www.htpcbeginner.com/access-usb-drive-in-virtualbox-guest-os/)
9. Configure Home Server on VirtualBox - [Ubuntu Server](http://www.htpcbeginner.com/ubuntu-home-server-on-virtualbox/)

## Downloading / Installing AtoMiC ToolKit 2.0

Downloading and using AtoMiC ToolKit 2.0 is the easy part. Just issue the following commands in sequence and you should be ready to roll.

$ sudo apt-get -y install git-core

$ git clone https://github.com/htpcBeginner/AtoMiC-ToolKit ~/AtoMiC-ToolKit

$ cd ~/AtoMiC-ToolKit

$ sudo bash setup.sh

Just choose the option you want to pursue and sit back, relax, and watch the magic that happens in front of you. You may come back and run the AtoMiC script any time you want to do any operation. Watch it in action in the video below.

**Install Syncthing for syncing files across multiple servers.**

# How To Install and Configure Syncthing to Synchronize Directories on Ubuntu 14.04

### Introduction

There are many programs capable of keeping files in sync between different computers. **Syncthing** is a compelling new option that is cross platform, completely open source, very flexible, and easy to use.

In this guide, we will show you how to get started using Syncthing to synchronize data between two Ubuntu 14.04 server instances. This will allow you to seamlessly sync files between these servers based on configured "watch" directories.

## Prerequisites and Goals

To complete this guide, you will need access to two Ubuntu 14.04 server instances.

For the purposes of this guide, we will call these servers:

* **serverone**
* **servertwo**

These will be equal partners, so there is not a primary/secondary relationship. You should have a regular user account configured on each of these (you can find out how by following steps 1-4 in our [initial server setup guide for Ubuntu 14.04](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-14-04) here).

The regular user account in this guide will be called demouser, but you should use whatever username you would like. This does not have to be separate from your regular non-root account.

We will install and configure Syncthing on each of these machines so that files can be shared between them. After you have completed the procedures outlined in this guide, you should have two servers, each with a directory that will mirror changes to the other computer.

We will install Syncthing as a system-level application and create an Upstart script. This will allow our Syncthing instances to start at boot and restart correctly, instead of spawning new instances each time as it does by default.

# Debian/Ubuntu Packages

The release channel is updated with full release builds, approximately once a week.

# Add the release PGP keys:

**curl -s https://syncthing.net/release-key.txt | sudo apt-key add -**

# Add the "release" channel to your APT sources:

**echo "deb http://apt.syncthing.net/ syncthing release" | sudo tee /etc/apt/sources.list.d/syncthing.list**

# Update and install syncthing:

**sudo apt-get update**

**sudo apt-get install syncthing**

## Change the GUI Options to Allow Remote Viewing

By default, the Syncthing web interface is only available to connections from the same computer. This will not work for our purposes since we are working with remote servers.

To fix this, we need to edit a line in the Syncthing configuration file. However, the configuration file has not been created yet.

To create the file automatically, we can start up the service briefly. Type the syncthing command to get this up and running:

syncthing

The syncthing process is not a daemon, which means that it runs in the current shell session (we will get around this by implementing an Upstart script momentarily). It will start the process and create some necessary files.

After a moment, you will see an informational message about your Node ID that looks like this:

[2EQK3] 15:47:15 OK: Ready to synchronize default (read-write)

[2EQK3] 15:47:15 INFO: Node 2EQK3ZR77PTBQGM44KE7VQIQG7ICXJDEOK34TO3SWOVMUL4QFBHA is "server1" at [dynamic]

When you see this, it means that the process has initialized correctly. We can now stop the process by typing:

CTRL-C

This will stop the process and give us control over the terminal again.

We can now edit the configuration file that was created. Open the file in your text editor:

nano ~/.config/syncthing/config.xml

Look for a section that deals with the GUI. It should look like this:

<gui enabled="true" tls="false">

<address>127.0.0.1:8080</address>

</gui>

The only change that we need to make is to replace the localhost address (127.0.0.1) with 0.0.0.0, which represents all network interfaces. This section will look like this when you are finished:

<gui enabled="true" tls="false">

<address>0.0.0.0:8080</address>

</gui>

Save and close the file when you are finished. We will do more extensive configuration later on through the web interface.

Again, complete these procedures on both of the servers you are configuring.

## Create an Upstart Script to Handle Service States

Next, we will implement an Upstart script to manage our process. This will allow us to automatically start the Syncthing process when the server boots. It will also let us run it as a service, without taking control of our session.

The Upstart script that we will use is taken from a [post on the Syncthing discussion boards](http://discourse.syncthing.net/t/keeping-syncthing-running/30). Create and open the Upstart file with root privileges in your editor by typing:

sudo nano /etc/init/syncthing.conf

Inside, we will use the following lines to control our Upstart process:

description "Syncthing P2P sync service"

start on (local-filesystems and net-device-up IFACE!=lo)

stop on runlevel [!2345]

env STNORESTART=yes

env HOME=/home/demouser

setuid "demouser"

setgid "demouser"

exec /usr/local/bin/syncthing

respawn

Let's go over what this means.

We start with a general description of the service. Next, we define when the service should start and stop. Here, we are telling the system that it should be started after the filesystem and the non-local networking interfaces are brought up. It will be stopped whenever the system leaves one of the normal runlevels.

Next, we will set an environmental variable that tells Syncthing not to restart using its own mechanisms. This is because it actually spawns a new process each time without cleaning up the old one. To avoid this, we will implement the ability to restart within this Upstart script later on.

The next environmental variable sets the home directory that will be used by Syncthing. This will be used to find the correct configuration files and pre-populate fields in the GUI. We will also specify the user and group using the setuid and setgid parameters respectively. Modify all three of these values to point to your user's information.

Next, we set the path to the actual command that we are going to execute. Finally, we use respawn to tell Upstart to automatically restart the process if it is stopped prematurely. This is used to automatically restart the process (cleanly) when the Syncthing process tries to restart itself internally.

When you are finished, save and close the file.

You can now start the service by typing:

sudo initctl start syncthing

Complete the above steps on both of your servers.

## Setting Up Security in the Web UI

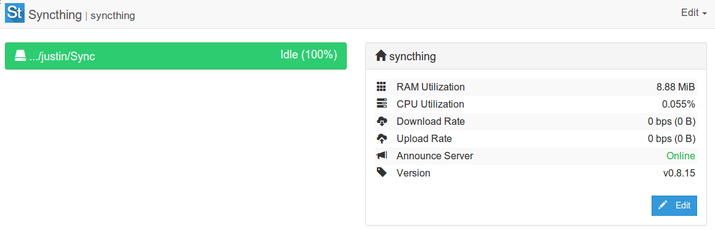
We are finally ready to configure our instances through the web user interface.

Log into each of your servers' Syncthing interfaces by visiting their public IP addresses and port 8080:

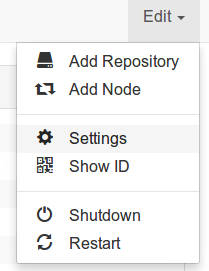
http://server1\_public\_IP:8080

http://server2\_public\_IP:8080

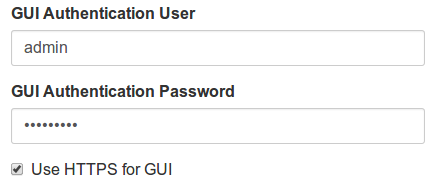
You will see the main Syncthing screen:



The first thing that we need to do is add some security to our interface. Go to the "Edit" menu in the upper right-hand corner and click on the "Settings" selection:

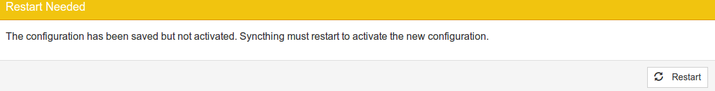


On the right side, we need to set up a username and password for the web interface. We also want to enable TLS encryption for our sessions so that our traffic cannot be intercepted in transit by checking the "HTTPS" box:



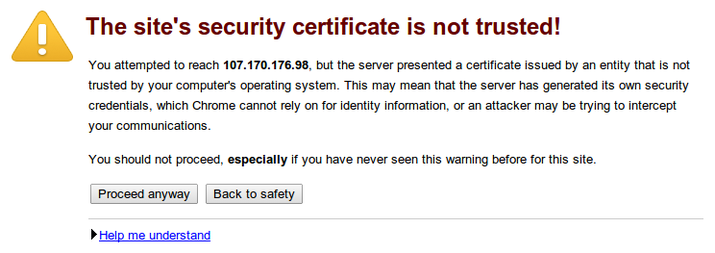
When you are finished, click the "Save" button at the bottom.

You will see a message that prompts you to restart the service for the changes to take affect:



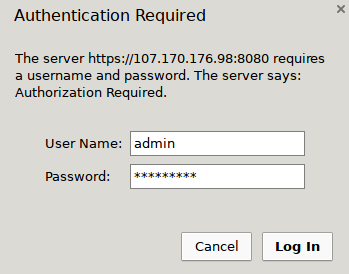
Click on the "Restart" button.

You will probably need to refresh the page for the changes to reload properly. When you do so, you will most likely encounter an SSL warning that looks like this:



This is just letting you know that the entity that signed the SSL certificate is not in your browser's list of trusted certificate authorities. This is expected since the certificates are self-signed. It is safe to click on "Proceed anyway" to continue.

You will then get an authentication prompt for the username and password that you configured:



Fill out the appropriate values and log in. Your interface is now more secure from outside traffic.

Complete these steps on both servers.

## Connecting the Two Servers and Sharing Directories

In order to share content, Syncthing requires both ends of the connection to add the other server to their list of nodes. Then, both sides must also add a repository (directory) to synchronize.

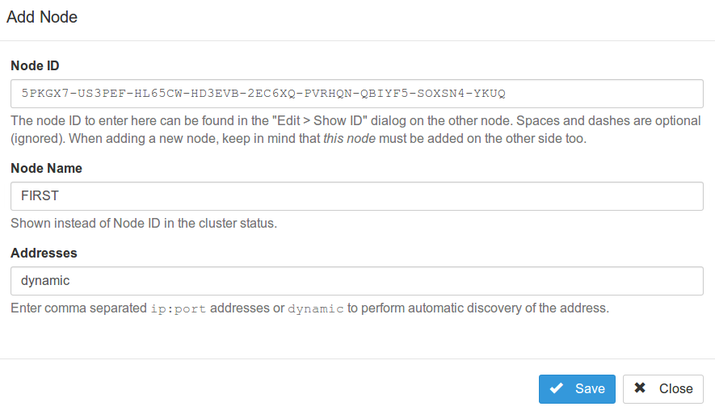
To add a node, you will need the companion node's ID. You can get this ID by going to the "Edit" menu in the upper-right corner and selecting the "Show ID" option.

This will bring up an overlay with a long ID that you can copy. It will also give you a QR code in case you are configuring smart phone application. Copy the ID of the first node.



On the second server, click on the "Edit" menu and select "Add Node".

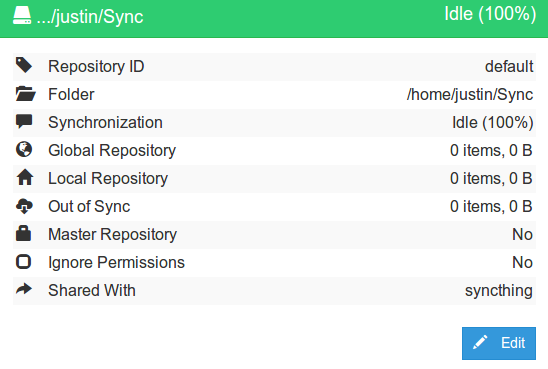
An overlay will appear with fields that you must configure to add the other server. In the "Node ID" field, paste the ID that you copied from the first server. In the "Node Name" field, select any descriptive name that you would like to use for your other server. Leave the "Addresses" as "dynamic":



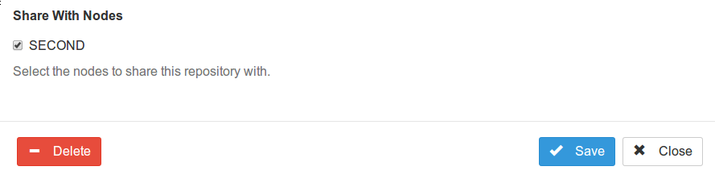
When you are finished, click "Save" to add the new node.

Run through this same process in the reverse order so that both of the servers have the opposite server in their configuration. For the moment, ignore any requests to restart the service, because we will be making a few more additional changes first.

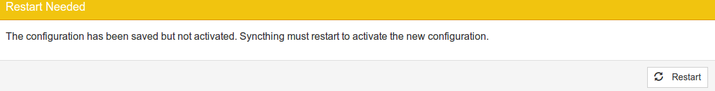
In the web interface, the nodes that you are connected to are shown on the right side. The repositories you are sharing are listed on the left. By default, a folder called Sync will be created in your home directory to act as the default repository. If the menu is compressed, click the name to expand the listing:



Click on the "Edit" button to configure the repo. Towards the bottom, you will have the option to share this repository with any of the nodes that you have configured. Check the box associated with the opposite node and then click the "Save" button:



Now, you can click on the "Restart" button to implement your changes:



Now, any changes made in the ~/Sync directory will be mirrored to the opposite server. By default, this will sync every 60 seconds, but this can be changed in the "Settings" menu.

You can add additional directories that you wish to mirror by choosing the "Add Repository" option from the "Edit" menu. The "Repository ID" must be the same on each of the servers for the directories to be associated.

## Conclusion

At this point, you should have two servers connected and syncing a single directory's contents. You can easily expand this by either adding additional repositories (directories), or by adding additional nodes to also sync with.

If you plan on syncing system files to directories that a normal user would not have write access to, you will probably need to modify the Upstart file to use the root account or whatever other user account necessary to perform the required actions.

**Using Amazon Cloud Drive with Plex Media Server on Ubuntu… And Encrypting It!**

[OCTOBER 19, 2016](https://enztv.wordpress.com/2016/10/19/using-amazon-cloud-drive-with-plex-media-server-and-encrypting-it/) / [ENZTV](https://enztv.wordpress.com/author/enztv/)

**UPDATED – 24October2016**

The idea of an “unlimited” Plex server is an exciting one, but there are some bases you’ll want to cover before dumping all your media into the cloud. The biggest issue with keeping your media hosted in the cloud is encryption; you don’t want any storage providers snooping around your files, regardless of their source or content. This guide will get you going, with a focus on security.

Also, with the [announcement of Plex’s new “Plex Cloud” offering](https://www.plex.tv/blog/book-plex-volume-3-plex-cloud/), I’ve seen quite a few people posting on [Reddit’s Plex subreddit](https://reddit.com/r/plex) for help with using [Amazon Cloud Drive](https://www.amazon.com/clouddrive/home) (ACD) as a storage provider for their Plex media on their standalone Plex server. There have been some guides floating around the Internet, but some of these guides use a confusing approach (in my opinion) to accomplish a task that’s already a bit confusing for some. I’ve been using the setup detailed in this guide for a few years now, even with multiple servers utilizing the same ACD mount, and I’ve had very few issues. I’ve automated this setup with tools like [CouchPotato](https://couchpota.to/), [Sonarr](https://sonarr.tv/), and custom scripts in conjunction with [Plex Requests](http://plexrequests.8bits.ca/) to the point where I rarely have to do any maintenance on my Plex servers, and anybody can request media and have it available in minutes without any kind of interaction from me.

If you haven’t set up Plex yet, or you’ve got a fresh installation of Ubuntu, you can use my other guide, [Getting Started with Plex](https://enztv.wordpress.com/2016/10/19/getting-started-with-plex/) to get the prerequisite setup finished first.

In this guide, we’ll be making use of several different projects to get this setup working smoothly, so while a guide can help you get set up, troubleshooting any problems can be difficult and a guide won’t always be helpful. However, if everything is set up in a logical way to begin with, the pain of troubleshooting can eased. Let’s get started on that now!

**What you’ll need:**

* Amazon Cloud Drive account (obviously)
* User account running these tools and Plex (preferably named “Plex”)
* unzip
* Python 3 (installed by default on Ubuntu 14.04+)

**What we’ll be installing:**

* [acd\_cli](https://github.com/yadayada/acd_cli)
* [EncFS](http://www.arg0.net/encfs)
* [UnionFS-Fuse](http://manpages.ubuntu.com/manpages/precise/man8/unionfs-fuse.8.html)
* [rclone](http://rclone.org/)

**How it Will Work:**

Basically, we will be using acd\_cli to **read** our Amazon Cloud Drive, rclone to **write** to ACD, and EncFS to **encrypt** it all. The real magic happens with UnionFS, as it will **merge** our read and write directories into one, so we can essentially forget about those other directories and focus on a single path for all our needs, including other tools like CouchPotato and Sonarr.

I’m going to assume your user account running Plex ***is not*** root, and is called “Plex” for the sake of convenience. Substitute usernames as necessary here, and stay logged in as this user throughout the guide. We’ll use “sudo” for anything requiring elevated privileges.

Anytime “~” is specified in a directory, it is assumed to be equal to ‘/home/plex’

**Step 1 – Creating Necessary Directories**

Let’s create the directories we’ll be using to make all this work. For ease of use, I like to put these directories right inside my Plex user’s home directory. So let’s make sure we’re at the right place (***again, you should be logged in as your Plex user, not root***):

cd ~

Now that we’re in the home directory, let’s create the five directories we’ll be using:

mkdir .acd

mkdir acd

mkdir .local

mkdir local

mkdir media

These directories will be used as follows:

~/.acd – ACD mount containing encrypted media  
~/acd – ACD mount decrypted with EncFS  
~/.local – local media after being encrypted by EncFS  
~/local – where your local/downlaoded media will be copied to  
~/media – the UnionFS mount point which will merge ‘~/acd’ and ‘~/local’ directories

**Step 2 – Setting Up acd\_cli**

Okay so let’s assume you’ve got your Amazon Cloud Drive (ACD) set up on Amazon.com. While you’re in that web portal, let’s create a folder called “Plex” since you might want to take advantage of that unlimited storage and keep other things in there besides your Plex media. I’m a sucker for organization, after all.

Now, hop over to your server (and open a terminal if you’re not using SSH). Make sure everything is ready to go for this setup by updating APT:

sudo apt-get update

**Install acd\_cli**

Now, the easiest way to get acd\_cli up and running is to just grab it with Pip:

sudo pip3 install --upgrade git+https://github.com/yadayada/acd\_cli.git

If you don’t have Pip installed, grab it with:

sudo apt-get install -y python3-pip

The GitHub repo for this project is updated roughly every other week, so you could create a cron job for keeping acd\_cli up to date, but honestly in most cases with a tool like this, unless something is broken, you probably don’t need to update. The only reason I would say upgrading is necessary is if the author made a huge improvement to the tool.

**Authorize acd\_cli**

Now that we have acd\_cli installed, we have to authorize the tool to allow access to your ACD.

***Using a desktop interface/GUI***

If you’re using a terminal in a desktop session (GUI), then simply enter the command below, otherwise scroll down a tad further to “Using SSH or a headless server”

acdcli init

This will open your browser to an ACD login page. Go ahead and enter your credentials. After that, your browser should download a file called “oauth\_data” automatically. This file will need placed in the ‘.cache’ directory inside your Plex user’s home directory. So, go ahead and hop over to your terminal and enter:

mv ~/Downloads/oauth\_data ~/.cache/acd\_cli/

You can just drag-and-drop that file from the browser right into the terminal window if it didn’t download to the default directory.

***Using SSH or a headless server***

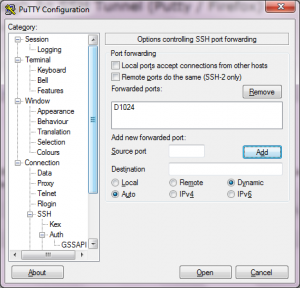
**My Experience Is that the text based browser does not work due to a captcha. The following steps worked for me.**

## Getting Started

First things first, make sure you have [Putty](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) and [Firefox](http://www.firefox.com/) installed.  Next you’ll want to be able to establish an SSH connection to a remote server using Putty.  For example, launch putty, enter the host name or IP and make sure you can connect and login.  Once you have that, you’re ready to setup the SSH tunnel and browse through it.

## Configuring The SSH Tunnel

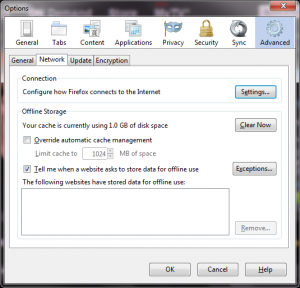
Now open Putty and Enter the hostname or IP of the machine you want to establish a remote connection to.  Next under Connection->SSH->Tunnels find the radio boxes under the Destination field and make sure Dynamic is selected.  Next under where it reads Add new forwarded port:enter a source port.  For this example lets use 1024, enter this in the source port field and click the Add button.  You should see a value in the Forworded ports: list that reads D1024.  That’s all there is to the Putty side of things.  Next go back to the session area and save the current configuration as a saved session if you’d like, then Open the SSH connection.

[](http://www.godlikemouse.com/wp-content/uploads/2011/08/putty.png)

*Configure putty*

## Configuring Firefox

## Now, launch Firefox, select Tools->Options and click the Advanced tab. Within the Advanced tab, click on the Network tab and click the Settingsbutton.

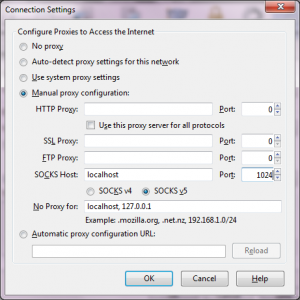
[](http://www.godlikemouse.com/wp-content/uploads/2011/08/options.png)

*Network Options*

Within the Network Settings dialog, select the Manual proxy configurationradio button and enter the following for the SOCKS Host: and Port:

SOCKS Host: localhost

SOCKS Port:1024

[](http://www.godlikemouse.com/wp-content/uploads/2011/08/connection_settings.png)

*Connection Settings*

Click OK on the Settings dialog, then Click OK on the Options dialog. Now you should be good to go.  Enter a new URL in the Firefox address bar and you’ll be browsing from the remote end of the SSH connection.

**Now you can follow the steps for the headless server**

**Copy the link shown when you run “acdcli init” and past into firefox.**

If you’re managing a headless Plex server, or you don’t have a desktop environment installed, authenticating acd\_cli gets a bit trickier. I’ve found success using the w3m browser. So lets install that:

sudo apt-get install -y w3m

This will install a text-only terminal-based browser. It should do fine for this purpose, but don’t plan on surfing the web with it. Now, go ahead and run the acd\_cli authorization command:

acdcli init

Then hit “CTRL+C” to cancel the action. w3m won’t automatically open, but all we need is the link acd\_cli was trying to open. So go ahead and copy that link, then paste it after entering “w3m” in the terminal, like so:

w3m https://tensile-runway-92512.appspot.com/

Using w3m is pretty simple, most clickable links/buttons show up as red text. So just click or use your arrow keys to put the cursor in the username field and press your “enter” key to start typing, then press the “enter” key again to complete that field. Do the same for the password field, then click or arrow over to the “Sign in using our secure server” link and press enter. You should see a bunch of text with “access-token” and some other text. This is your oauth\_data token which acd\_cli needs. Now just press “ESC+S” to save this file, and specify the path as “~/.cache/acd\_cli/oauth\_data” to save it where it needs to go.

**Easiest way to get the downloaded oauth file to the server is to use the following commands:**

Sudo vi ~/.cache/acd\_cli/oauth\_data

now open the oauth file that was downloaded in a text editor such as notepad. Copy the data contained in the file and paste into vi.

Then

Esc

:wq

Regardless of the method you used to get your oauth\_data file, it’s a good idea to keep this backed up somewhere outside the “~/.cache/acd\_cli” folder just in case.

**Mounting ACD**

Now we’re ready to make your ACD accessible to your server. First thing we’ll do is synchronize acd\_cli to make sure it’s connected to your account properly:

acdcli sync

If this command completes successfully (no output errors), then we can move on. If you get any errors, the content of the error should point you in the right direction of the root cause. Either acdcli isn’t installed properly, or the oauth\_data file didn’t download properly.

So, let’s go ahead and mount your ACD to the (soon-to-be) encrypted ACD directory:

acdcli mount -ro /home/plex/.acd

Now if you “cd” into ‘~/.acd’ you should see the “Plex” directory you created in the ACD web portal way back at the beginning of this guide. This means your ACD is now mounted on your server! That’s about it for acd\_cli, we have our ACD mounted and visible to our server, so lets move on to encrypting it!

**Step 3 – Setting up EncFS**

**Install EncFS**

First thing’s first; let’s make sure EncFS is installed:

sudo apt-get install -y encfs

**Set Up and Test Encryption**

Then we’ll go ahead and get our directories encrypted. Using EncFS, the first directory in the command is the one you’ll want encrypted, and the second directory is the one you want decrypted. The nice thing about EncFS is you don’t really have to specify the action, if the first directory already contains encrypted files, the second directory will automatically decrypt them. So let’s encrypt our local media directories:

encfs /home/plex/.local /home/plex/local

You should see a prompt from EncFS regarding encryption settings. Use the default and just press “enter” to move on to password creation. **Make sure you remember this password!**Type it a second time, and your local directories are now encrypted. You can test this by creating the directories Plex will use in the ‘/home/plex/local’ directory. In this guide, we’ll only worry about movies and TV shows, so let’s create a directory for each:

cd ~/local

mkdir Movies

mkdir "TV Shows"

Now let’s list the ‘local’ directory to make sure they’re created:

ls ~/local

And you should see the “movie” and “tv” directories listed. Now, let’s list the encrypted directory:

ls ~/.local

You should see two directories with a bunch of random characters. This confirms our files are being encrypted.

**Move Settings File**

Now let’s move our EncFS settings file to a more convenient location. First we need to unmount our local media directories. This command is also used to unmount other EncFS directories in the future, though you shouldn’t need to use it very often:

fusermount -u /home/plex/local

Next, we’ll move the “encfs6.xml” file and rename it to something simpler:

mv /home/plex/.local/.encfs6.xml  /home/plex/encfs.xml

**Encrypting Directories Using EncFS Configuration Files**

This “encfs.xml” file is your encryption “key,” not to be confused with the password. These work in tandem to keep your files secure, so you’ll want to make sure you keep a copy of this file backed up somewhere safe. Now, when mounting an EncFS directory, you’ll need to specify the location of the config file. Anytime you mount an encrypted directory, the command will always start the same, only the directories will change as needed. Let’s go ahead and mount our local media directory again using the config file:

ENCFS6\_CONFIG='/home/plex/encfs.xml' encfs /home/plex/.local /home/plex/local

Okay, now let’s do the same for our ACD directories. We’ll be adding ‘/Plex’ onto the ‘.acd’ directory because we added that folder to ACD for organization purposes, and only the files inside that folder need encrypted. So enter the following into your terminal:

ENCFS6\_CONFIG='/home/plex/encfs.xml' encfs /home/plex/.acd/Plex /home/plex/acd

And that’s it for encryption! It seems like a lot, but underneath it all, everything is pretty simple. Four directories, two of which are encrypted mirrors of the others, and a fifth to merge the two decrypted directories together. Let’s move on to merging those directories with UnionFS!

**Step 4 – Merging directories using UnionFS-Fuse**

**Install UnionFS-Fuse**

Once again, let’s get UnionFS-Fuse installed;

sudo apt-get install -y unionfs-fuse

**Merge Directories Using UnionFS**

There’s no other setup required for this part, we just need to specify which directories we want to merge. Those will be the decrypted directories, local and ACD, so Plex can be refreshed to grab new movies/episodes when they’re downloaded without waiting for them to be uploaded, and so other tools like CouchPotato and Sonarr can move files into this directory and they’ll appear to be in the same location as all the ACD media.

unionfs-fuse -o cow /home/plex/local=RW:/home/plex/acd=RO /home/plex/media/

The “RW” and “RO” arguments in that command specify which directory is writable (local directory) and which is read-only (ACD directory). So now anything you copy/move to ‘/home/plex/media’ will actually be in ‘/home/plex/local’ and thus encrypted in ‘/home/plex/.local’, and anything in your ‘/home/plex/acd’ directory will also show up in the ‘/home/plex/media’ directory. We only need to worry about one directory to point Plex libraries at, or to have CouchPotato/Sonarr monitor. This makes life much, *much* easier! Now, we’re ready to move on to the final phase of this guide; uploading your downloaded media to ACD.

**Step 5 – Adding ACD Directories to Plex**

Let’s make sure Plex can see the ACD directories now! Hop over to your Plex web portal, and add a new library (or edit an existing one). When choosing your directory, browse to ‘/home/plex/media’ and you should see the “movie” and “tv” directories we created. Choose the respective directory for your library, and that’s it! Anything you upload to ACD will show up in that directory.

**If you don’t see the ‘media’ directory when trying to add it through Plex**, it could be a permissions issue. One thing you can try is browsing to the ‘/home/plex/acd’ directory from Plex web and seeing if those directories show up. If they do, this is a permissions issue with the UnionFS mount. Unmount it by typing:

fusermount -u /home/plex/media

And then edit  the fuse configuration file:

sudo nano /etc/fuse.conf

Find the line “#user\_allow\_other” and remove the “#” by pressing the delete key in front of that line. Save the file by pressing “CTRL+X” and “Y” to overwrite. Then, mount the merged UnionFS-Fuse directory by adding the “allow\_other” option to the mount command:

unionfs-fuse -o cow,allow\_other /home/plex/local=RW:/home/plex/acd=RO /home/plex/media/

Now, you should be able to see the “movie” and “tv” folders inside ‘/home/plex/media’ directory when adding a library in Plex web.

**Step 6 – Uploading to ACD with rclone**

Now that our directories are set up and encrypted and we have them added to our Plex server, it’s time to start adding media to them.

**Install rclone**

Once more, let’s install another tool! rclone is a bit tougher to get installed, it’s not as automatic as the other tools we’ve used. Start off by making sure you’re in your Downloads directory, then grabbing the rclone package:

cd ~/Downloads

wget http://downloads.rclone.org/rclone-current-linux-amd64.zip

Now, unzip the archive and change directories to it:

unzip rclone-current-linux-amd64.zip

cd rclone-\*-linux-amd64

Copy the binary and give root ownership:

sudo cp rclone /usr/sbin/

sudo chown root:root /usr/sbin/rclone

sudo chmod 755 /usr/sbin/rclone

Then install the manpage so you can read all about rclone:

sudo mkdir -p /usr/local/share/man/man1

sudo cp rclone.1 /usr/local/share/man/man1/

sudo mandb

**Set Up rclone**

Now we’re ready to configure rclone. Once again, this process is different depending on whether you’re using a desktop session or a headless SSH session. If you’re using a terminal in a desktop session (GUI), then simply enter the commands below, otherwise scroll down a tad further to “Using SSH or a headless server”

***Using a desktop interface/GUI***

In your terminal, simply type the following command to enter the rclone configuration process:

rclone config

This will guide you through a fairly easy process. Let’s go through the steps together:

* Press “n” to add a new remote connection, then “enter” to proceed
* You’ll want to name your instance something simple, like “acd” to make it easier to upload files
* Press “1” for ACD, then “enter” to proceed
* Press “enter” for the next two fields to use defaults
* Press “y” to use the “automatic” mode, which will open a browser window. Enter your ACD credentials, and log in. The config process should continue automatically.
* Press “y” if you can see your token text on the screen
* Press “q” to exit the config process, we’re done for now

***Using SSH or a headless server***

If you don’t have the luxury of using a browser, we can make use of the w3m browser we installed back when setting up acd\_cli. However, the authorization process for rclone works a bit differently. For the easiest method, we are going to install a neat utility (one you should probably always install anyways, to be honest) called Screen. Screen allows you to have multiple terminal sessions open without worrying about them expiring or closing out. This is useful if you need to access your server remotely with Putty or something similar to execute a script or other process, but can’t have the SSH session open until that script/process is complete. Ending your SSH session also ends whatever script/process you were running. Screen is a great way to prevent that from happening, and lets you reconnect to existing sessions from any other SSH session. So let’s install Screen:

sudo apt-get install -y screen

Sometimes Screen is already installed, but either way it should be now for sure. To open a new terminal session with Screen, just type:

screen

Press “enter” to dismiss the message, and now you’re in a second terminal session. You can exit a Screen session while leaving it active at anytime by pressing “CTRL+A” then “D” to disconnect. Now that we have a second session running, let’s start the rclone config process:

rclone config

Move past these steps the same as the GUI users would have, up to the “use auto-config” step:

* Press “n” to add a new remote connection, then “enter” to proceed
* You’ll want to name your instance something simple, like “acd” to make it easier to upload files
* Press “1” for ACD, then “enter” to proceed
* Press “enter” for the next two fields to use defaults
* PAUSE at the auto-config step. Pressing “y” to use the “automatic” mode will attempt to open a browser window. This won’t work, however it will show you a URL to use. Copy this URL, then disconnect this Screen session by pressing “CTRL+A” then “D” which will keep that session active in the background
* Open the w3m browser to the rclone URL by entering:

w3m http://127.0.0.1:53682/auth

* Enter your ACD credentials by moving your cursor to the required fields, pressing “enter” and typing your username/password, then pressing “enter” again
* Place your cursor on “Sign in using our secure server” and press “enter” to sign in. You should see a “success” message
* Press “q” to exit the w3m browser
* Type the following command to reconnect to your previous Screen session:

screen -r

* Press “y” if you can see your token text on the screen
* Press “q” to exit the config process, we’re done for now
* Press “CTRL+A” then “K” to permanently kill this Screen session

**Upload Local Files/Directories to ACD Using rclone**

The syntax for copying files with rclone is “rclone copy sourcedir remote:destinationdir”. In this context, “sourcedir” is your local encrypted directory, “remote” is what you named your ACD instance when setting up rclone (I suggested acd), and “destinationdir” is the directory in your ACD you want to move those files to, which in our case would be inside the “Plex” folder you created in the ACD web portal.

Anyways, on to the meat and potatoes! To upload your local files over to your ACD mount, we’ll be uploading your encrypted local directory to the encrypted ACD mount with the following command:

rclone copy /home/plex/.local acd:Plex

This should copy the files from your local encrypted directory to your encrypted ACD directory, and then you can manually delete them from ‘home/plex/local’ if the copy was successful.

One thing I like to do is upload my movie and TV directories separately by appending those directory names to the ends of the source and destination directories in the rclone move command. You can just list your ‘/home/plex/.local’ directory to get the names of the directories, and you can figure out which is which by using the “touch” command to place a test file in either directory, then listing the encrypted directories to see which has a file in it. Keep note of those directory names then for future reference, and then you can add them into your “rclone move” command rather than using the “rclone copy” command to have these sub-directories automatically deleted after a successful copy. Here’s what my commands look like to upload with rclone (using my encrypted directory names):

rclone copy /home/plex/.local/BZxQ7QeNh8jos9pEzcKXk5vy/ acd:Plex/BZxQ7QeNh8jos9pEzcKXk5vy

rclone copy /home/plex/.local/L,ULJSSLPjETMTzk5OhzCioC/ acd:Plex/L,ULJSSLPjETMTzk5OhzCioC

And then to empty out my local storage while maintaining those parent directories:

rm -Rf "/home/plex/local/Movies/"\*

rm -Rf "/home/plex/local/TV Shows/"\*

This will maintain a “Movies” and “TV Shows” directory inside the ‘/home/plex/local’ directory for you.

**Synchronizing acd\_cli to Show New Files**

After we upload files using rclone, we then have to re-synchronize our acd\_cli mount to show the updated directories/files.

acdcli sync

After running the sync command, list your ‘/home/plex/acd/movies’ or ‘/home/plex/acd/tv’ directory, and you should see the directory you just uploaded. List that directory to make sure the file is there, and that’s it! Force a library update in Plex, and your movie/episode should be picked up in a few seconds.

**Step 7 – Use Crontab to Mount acd-cli, EncFS, and UnionFS Mounts Automatically (UPDATE)**

I’ve added a new post for handling all your mounts automatically! Check out [Automatically Mounting acd\_cli, EncFS, and UnionFS Mounts at Boot](https://enztv.wordpress.com/2016/10/24/automatically-mounting-acd_cli-encfs-and-unionfs-mounts-at-boot/)

**Automatically Mounting acd\_cli, EncFS, and UnionFS Mounts at Boot**

[OCTOBER 24, 2016](https://enztv.wordpress.com/2016/10/24/automatically-mounting-acd_cli-encfs-and-unionfs-mounts-at-boot/) / [ENZTV](https://enztv.wordpress.com/author/enztv/)

People have been asking how to mount all those different directories we set up in [Using Amazon Cloud Drive with Plex Media Server on Ubuntu… And Encrypting It!](https://enztv.wordpress.com/2016/10/19/using-amazon-cloud-drive-with-plex-media-server-and-encrypting-it/) at boot automatically. So here’s a quick guide on getting that set up. As usual, we’ll be assuming you’re using a user called “plex” for all this, and the directory structure from the guide as well.

I’ve played around with using fstab to mount all these directories, but there are just too many hurdles and things that tend to go wrong, between acd\_cli needing the “\_netdev” option to wait for a network device, and the EncFS mounts trying to run before acd\_cli is fully mounted. Not only all that, but anything mounted automatically in fstab is mounted as root, which can cause another set of issues with accessing the directories as another user. I’ve found that a simple crontab entry is the easiest and most straightforward method.

**Step 1 – Create a Password File for EncFS**

To fully automate the mounting of our EncFS directories, we’ll have to have the password readily available to pass into our script. I like to keep a “scripts” directory right inside my plex home folder with another directory for logs, so let’s create that first and then move into that scripts directory:

cd ~

mkdir scripts

mkdir scripts/logs

cd scripts

And now we can create a password file. Just run:

nano encfspass

And type *only* your EncFS password. When you’re finished, press “CTRL+X” and “Y” to save the file. Make that file executable with the following command:

chmod +x encfspass

**Step 2 – Write a Script**

We’ll need a script that will automatically clear out any mount points (if any) and then mount all our different shares. Go ahead and start off with opening Nano with a filename of “mount.sh” like so:

nano mount.sh

And then copy and paste the following:

#!/bin/sh

/bin/fusermount -uz /home/plex/acd

/bin/fusermount -uz /home/plex/.acd

/bin/fusermount -uz /home/plex/local

/bin/fusermount -uz /home/plex/.local

/bin/fusermount -uz /home/plex/media

acdcli sync

acdcli mount -ro /home/plex/.acd

ENCFS6\_CONFIG='/home/plex/encfs.xml' encfs --extpass="cat /home/plex/scripts/encfspass" /home/plex/.acd/Plex /home/plex/acd

ENCFS6\_CONFIG='/home/plex/encfs.xml' encfs --extpass="cat /home/plex/scripts/encfspass" /home/plex/.local /home/plex/local

unionfs-fuse -o cow /home/plex/local=RW:/home/plex/acd=RO /home/plex/media

exit

And to make the script able to be executed, we have to add the “execute” permission to the file:

chmod +x mount.sh

Now that you’ve got your script written, we can add it to our cron list!

**Step 3 – Add Mount Script to Crontab**

Start by editing your”2 crontab jobs:

crontab -e

If this is your first time editing crontab, you’ll be asked to choose a text editor. If you haven’t noticed, I prefer Nano. Press “2” and “enter” to continue. Now, use your arrow keys to scroll down to the last blank line in the file, and enter the following line:

@reboot /home/plex/scripts/mount.sh > /home/plex/scripts/logs/mount.log 2>&1

“@reboot” obviously means this script will be run on a system reboot. The 2>&1 bit at the end will append to the specified log file, just in case something goes wrong with our script.

**Wrap-Up**

So that’s about it. The nice thing about this method is, if anything happens to your mounts or directories, you can just hop over to your scripts directory and run that mount.sh script. It should clear up most issues. Not having to worry about manually mounting anything during reboots is definitely a plus! As always, leave a comment if you have any issues with the guide, or [drop me a message on Reddit](https://www.reddit.com/user/enz1ey/). Thanks for reading!