

# **Free and Open Source Software Lab**

**International Centre for Free and Open Source Software (ICFOSS)**

---

Thejaswini, Technopark, Thiruvananthapuram-695581



# **CONTENTS**

<b>Experiments</b>
<b>1. Getting started with Linux basic commands and directory structure, execute file, directory operations.</b>
<b>2. Linux commands for redirection, pipes, filters, job control, file ownership, file permissions, links and file system hierarchy.</b>
<b>3. Shell Programming : Write shell script to show various system configuration like</b>  <b>3.1 Currently logged user and his logname</b> <b>3.2 Your current shell</b> <b>3.3 Your home directory</b> <b>3.4 Your operating system type</b> <b>3.5 Your current path setting</b> <b>3.6 Your current working directory</b> <b>3.7 Show Currently logged number of users</b>
<b>4. Write shell script to show various system configuration like</b>  <b>4.1 About your OS and version, release number, kernel version</b> <b>4.2 Show all available shells</b> <b>4.3 Show mouse settings</b> <b>4.4 Show computer CPU information like processor type, speed etc</b> <b>4.5 Show memory information</b> <b>4.6 Show hard disk information like size of hard-disk, cache memory, model etc</b>  <b>4.7 File system (Mounted)</b>
<b>5. Shell script program for scientific calculator.</b>
<b>7. Version Control System setup and usage using GIT.</b>  <b>7.1 Creating a repository</b> <b>7.2 Checking out a repository</b> <b>7.3 Adding content to the repository</b> <b>7.4 Committing the data to a repository</b>
<b>9. Shell script to implement a script which kills every process which uses more than a specified value of memory or CPU and is run upon system start.</b>
<b>11. Running PHP : simple applications like login forms after</b>

<b>12. Advanced linux commands curl, wget, ftp, ssh and grep</b>
<b>13. Application deployment on a cloud-based LAMP stack/server with PHP eg: Openshift, Linode etc.</b>
<b>15. Virtualisation environment (e.g., xen, qemu, virtualbox or lguest) to test an applications, new kernels and isolate applications. It could also be used to expose students to other alternate OSs like *BSD</b>
<b>17. Introduction to packet management system : Given a set of RPM or DEB, how to build and maintain, serve packages over http or ftp. and also how do you configure client systems to access the package repository.</b>
<b>18. Installing various software packages. Either the package is yet to be installed or an older version is existing. The student can practice installing the latest version. Of course, this might need Internet access.</b>  <b>18.1 Install samba and share files to windows</b> <b>18.2 Install Common Unix Printing System(CUPS)</b>

# **1. Getting started with Linux basic commands and directory structure, execute file, directory operations.**

## **Summary**

Free and open source software gives the user the freedom to use, copy, distribute, examine, change and improve the software. These rights are stipulated in licenses. Development takes place within the form of projects, and those who participate are mostly professional software developers who either take part on paid time with the permission of their employer, or on their own time. Many large suppliers (IBM, HP, Sun, etc) use free and open source software in their own products and solutions and also contribute with their own resources in the various projects. Free and open source software entails a new kind of competition, separated from that of traditional business in that the product generally is not owned by any single company and therefore cannot be purchased of the market. Furthermore, the software itself is not constricted by any cost or fee. It can be obtained free of charge on the Internet. The conclusion of the working group is that free and open source software in many ways, both functionally and qualitatively, is quite equivalent to – or better than – proprietary products. Free and open source software should therefore be judged on an even par with proprietary software in a procurement process in order to establish better market competition. It is also necessary to place demands on open standards and file formats in order to achieve interoperability between different systems. Proprietary software is mainly sold nowadays without any guarantees, and producers exclude responsibility for, more or less, all errors and bugs in the product. The producer dictates conditions for changes and corrections, and the lifetime of a product is usually short before new versions are released. If the software producer goes out of business, all development and support disappear and the user has to look for alternative solutions. With free and open source software these dependencies can be avoided. The concept of free and open source software has existed for almost twenty years, and Linux has been around for about ten. Many of today's Internet functions are almost completely based on free and open source software, functions such as e-mail and the translation of computer names to IP addresses, i.e. DNS. More than 65% of all web servers on the Internet are based on open source software.

## **Free and Open Source Software (FOSS)**

### **What is Free and Open Source Software?**

One could illustrate the idea of free and open source software compared to proprietary software with a situation where a person buys a house. Upon purchasing the house he also receives technical drawings and blueprints so that he can make changes himself to the structure. One alternative would be for the seller to retain ownership. All drawings would remain

in his possession and therefore all changes to the house would have to be made by the same vendor or seller. Another case could be the purchase of a car, where only the dealer's repair shop retains all car manuals and repair guides for exclusive rights to repair or make any changes in the performance of the car.

Richard M Stallman, founder of Free Software Foundation, uses the illustration of cookery and food. How would we experience the world around us if recipes were not freely available or free to change and modify? What would it be like if we committed a crime every time we made a copy of the recipe or gave it to someone else? Free and open source software means that source code is freely accessible, that the software can be freely used, changed, improved, copied and distributed by all who wish to do so.

Free and open source software does not have to be free of charge. Besides being able to construct business models around the software based on commercial aspects, a company can receive direct payment by the use of a large number of licensing schemes and models. These models can also be included in the overall definition of what we mean by free and open source software. What is important here is that the source code is available to the customer.

## **Why Free and Open Source Software?**

One of the strongest arguments for using free and open source software is the opportunity to arrive at a higher degree of independence regarding price and licensing conditions. In a situation with economic restraint, new and more rigorous licensing conditions, and with software that becomes replaced by newer versions more often than before, the software environment becomes increasingly more expensive. Free and open source software enjoys a significant market share in many areas. Notably, Apache is used for more than 65% of all Internet web servers, often with Linux as operating system. Free and open source software often has higher dependability, and in many cases better performance when directly compared to its proprietary counterpart. Scalability and flexibility within the model for the development of free and open source software enables it to be developed for a large number of platforms and environments. An area difficult to measure and compare is security, but it has been found that it is just as good, if not better, than proprietary alternatives. Free and open source software is less prone to attacks and virus over the Internet. As far as costs are concerned, it is to the advantage of free and open source software, especially if one looks exclusively at direct costs.

One point that is mentioned in different studies is the risk of becoming dependent on one vendor who in turn can control the whole infrastructure. In Germany, the use of free and open source software is motivated among other things by the need for security. The German minister of the interior has expressed that "monoculture is not good for security".

# Applications

## Office and Productivity Tools

Most computers are probably used for “office applications” like writing letters and memos, seminar papers or Ph. D. theses, the evaluation of data using spreadsheets and graphics packages and similar jobs. Users also spend large parts of their computer time on the Internet or reading or writing e-mail. No wonder that there is a lot of good free software to help with this. Most of the programs in this section aren’t just for Linux, but are also available for Windows, OS X or even other Unix variants. This makes it possible to gradually move users into a FOSS ecosystem by installing, e. g., LibreOffice, Firefox, and Thunderbird on a Windows PC in place of Microsoft Office, Internet Explorer, and Outlook, before replacing the operating system itself with Linux. If you play your cards right <sup>6</sup>, the users might not even notice the difference.

**OpenOffice.org** has been the FOSS community’s flagship big office-type application for years. It started many years ago as “StarOffice” and was eventually acquired by Sun and made available as free software (in a slightly slimmed-down package). OpenOffice.org contains everything one would expect from an office package—a word processor, spreadsheet, presentation graphics program, business chart generator, database, and can also handle (in a fashion) the file formats of its big competitor from Microsoft.

**LibreOffice** After Sun was taken over by Oracle and the future of OpenOffice.org was up in the air, some of the main OpenOffice.org developers banded together and published their own version of OpenOffice.org under the name of “LibreOffice”. Both packages are currently maintained side by side (Oracle donated OpenOffice.org to the Apache Software Foundation) not the optimal state, but it is unclear whether and how there will be a “reunification”.

By now, most major Linux distributions contain LibreOffice, which is more aggressively developed and—more importantly—cleaned up.

**Firefox** is by now the most popular web browser and is being distributed by the Mozilla Foundation. Firefox is more secure and efficient than the former “top dog” (Microsoft’s Internet Explorer), does more and conforms better to the standards governing the World Wide Web. In addition, there is a large choice of extensions with which you can customise Firefox according to your own requirements.

**Chromium** is the FOSS variant of the Google browser, Chrome. Chrome has recently started competing with Firefox—Chrom{e,ium} is also a powerful, secure browser with various extensions. Especially Google’s web offerings are geared towards the Google browser and support it very well.

**Thunderbird** is an e-mail program by the Mozilla Foundation. It shares large parts of its underlying infrastructure with the Firefox browser and—like Firefox—offers a large pool of extensions for various purposes.

## **Graphics and Multimedia Tools**

Graphics and multimedia has long been the domain of Macintosh computers (even though the Windows side of things has to offer nice software, too). Admittedly, Linux is still missing true equivalents to programs like Adobe's Photoshop, but the existing software is also nothing to scoff at.

**The GIMP** is a program for editing photographs and similar images. It is not quite up to par with Photoshop (for example, some pre-press functionality is missing) but is absolutely usable for many purposes and even offers a few tools for jobs such as creating graphics for the World Wide Web that Photoshop does not do as conveniently.

**Inkscape** If The GIMP is Linux's Photoshop, then Inkscape corresponds to Illustrator a powerful tool for creating vector-based graphics.

**ImageMagick** is a software package that allows you to convert almost any graphic format into nearly every other one. It also enables script-controlled manipulation of images in endless different ways. It is wonderful for web servers and other environments where graphics need to be processed without a mouse and monitor.

**Audacity** serves as a multi-track recorder, mixing desk and editing station for audio data of all kinds and is also popular on Windows and the Mac.

**Cinelerra** and other programs like KDenlive or OpenShot are "non-linear video editors" that can edit and dub video from digital camcorders, TV receivers, or webcams, apply various effects and output the result in various formats (from YouTube to DVD).

**Blender** is not just a powerful video editor, but also allows photorealistic "rendering" of three-dimensional animated scenes and is hence the tool of choice for creating professional-quality animated films. We might just as well mention here that today no Hollywood blockbuster movie is produced without Linux. The big studios' special effects "render farms" are now all based on Linux.

## **Internet Services**

Without Linux, the Internet would not be recognisable: Google's hundreds of thousands of servers run Linux just as the trading systems of most of the world's big stock exchanges (the German exchange as well as the London and New York stock exchanges), since the required

performance can only be made available with Linux. In point of fact, most Internet software is developed on Linux first, and most university research in these areas takes place on the open-source Linux platform.

**Apache** is by far the most popular web server on the Internet more than half of all web sites run on an Apache server. There are of course other good web servers for Linux—for example, Nginx or Lighttpd—, but Apache remains the most common.

**MySQL and PostgreSQL** are freely available relational database servers. MySQL is best used for web sites, while PostgreSQL is an innovative and high performance database server for all kinds of purposes.

**Postfix** is a secure and extremely powerful mail server which is useful for any environment from the “home office” to large ISPs or Fortune 500 enterprises.

### **Infrastructure Software**

A Linux server can prove very useful within a local-area network: It is so reliable, fast, and low-maintenance that it can be installed and forgotten (except for regular backups, of course!).

**Samba** turns a Linux machine into a server for Windows clients which makes disk space and printers available to all Windows machines on the network (Linux machines, too). With the new Samba 4, a Linux server can even serve as an Active Directory domain controller, making a Windows server extraneous. Reliability, efficiency and saved licence fees are very convincing arguments.

**NFS** is the Unix environment to Samba and allows other Linux and Unix machines on the network access to a Linux server’s disks. Linux supports the modern NFSv4 with enhanced performance and security.

**OpenLDAP** serves as a directory service for medium and large networks and offers a large degree of redundancy and performance for queries and updates through its powerful features for the distribution and replication of data.

**DNS and DHCP** form part of the basic network infrastructure. With BIND, Linux supports the reference DNS server, and the ISC DHCP server can cooperate with BIND to provide clients with network parameters such as IP addresses even in very large networks. Dnsmasq is an easy-to-operate DNS and DHCP server for small networks.

### **Programming Languages and Development**

From its beginnings, Linux has always been a system by developers for developers. Accordingly, compilers and interpreters for all important programming languages are



available—the GNU compiler family, for example, supports C, C++, Objective C, Java, Fortran and Ada. Of course the popular scripting languages such as Perl, Python, Tcl/Tk, Ruby, Lua, or PHP are supported, and less common languages such as Lisp, Scheme, Haskell, Prolog, or Ocaml, are also part of many Linux distributions. A very rich set of editors and auxiliary tools makes software development a pleasure. The standard editor, vi is available as are professional development environments such as GNU Emacs or Eclipse.

Linux is also useful as a development environment for “embedded systems”, namely computers running inside consumer appliances that are either based on Linux itself or use specialised operating systems. On a Linux PC, it is straightforward to install a compiler environment that will generate machine code for, say, ARM processors. Linux also serves to develop software for Android smartphones, and professional tools for this purpose are supplied for free by Google.

## **Basic Commands**

cd	Changes a shell’s current working directory
cp	Copies files
find	Searches files matching certain given criteria
less	Displays texts (such as manual pages) by page
ln	Creates (“hard” or symbolic) links
locate	Finds files by name in a file name database
ls	Lists file information or directory contents
mkdir	Creates new directories
more	Displays text data by page
mv	Moves files to different directories or renames them
pwd	Displays the name of the current working directory
rm	Removes files or directories
rmdir	Removes (empty) directories

slocate            Searches file by name in a file name database, taking file permissions into account

updatedb        Creates the file name database for locate

xargs            Constructs command lines from its standard input

Directory	Content
/bin	Common programs, shared by the system, the system administrator and the users.
/boot	The startup files and the kernel, vmlinuz. In some recent distributions also grub data. Grub is the GRand Unified Boot loader and is an attempt to get rid of the many different boot-loaders we know today.
/dev	Contains references to all the CPU peripheral hardware, which are represented as files with special properties.
/etc	Most important system configuration files are in /etc, this directory contains data similar to those in the Control Panel in Windows
/home	Home directories of the common users.
/initrd	(on some distributions) Information for booting. Do not remove!
/lib	Library files, includes files for all kinds of programs needed by the system and the users.
/lost+found	Every partition has a lost+found in its upper directory. Files that were saved during failures are here.
/misc	For miscellaneous purposes.
/mnt	Standard mount point for external file systems, e.g. a CD-ROM or a digital camera.
/net	Standard mount point for entire remote file systems
/opt	Typically contains extra and third party software.
/proc	A virtual file system containing information about system resources. More information about the meaning of the files in proc is obtained by entering the

	command <b>man proc</b> in a terminal window. The fileproc.txt discusses the virtual file system in detail.
/root	The administrative user's home directory. Mind the difference between /, the root directory and /root, the home directory of the <i>root</i> user.
/sbin	Programs for use by the system and the system administrator.
/tmp	Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work!
/usr	Programs, libraries, documentation etc. for all user-related programs.
/var	Storage for all variable files and temporary files created by users, such as log files, the mail queue, the print spooler area, space for temporary storage of files downloaded from the Internet, or to keep an image of a CD before burning it.

## File & Directory Commands

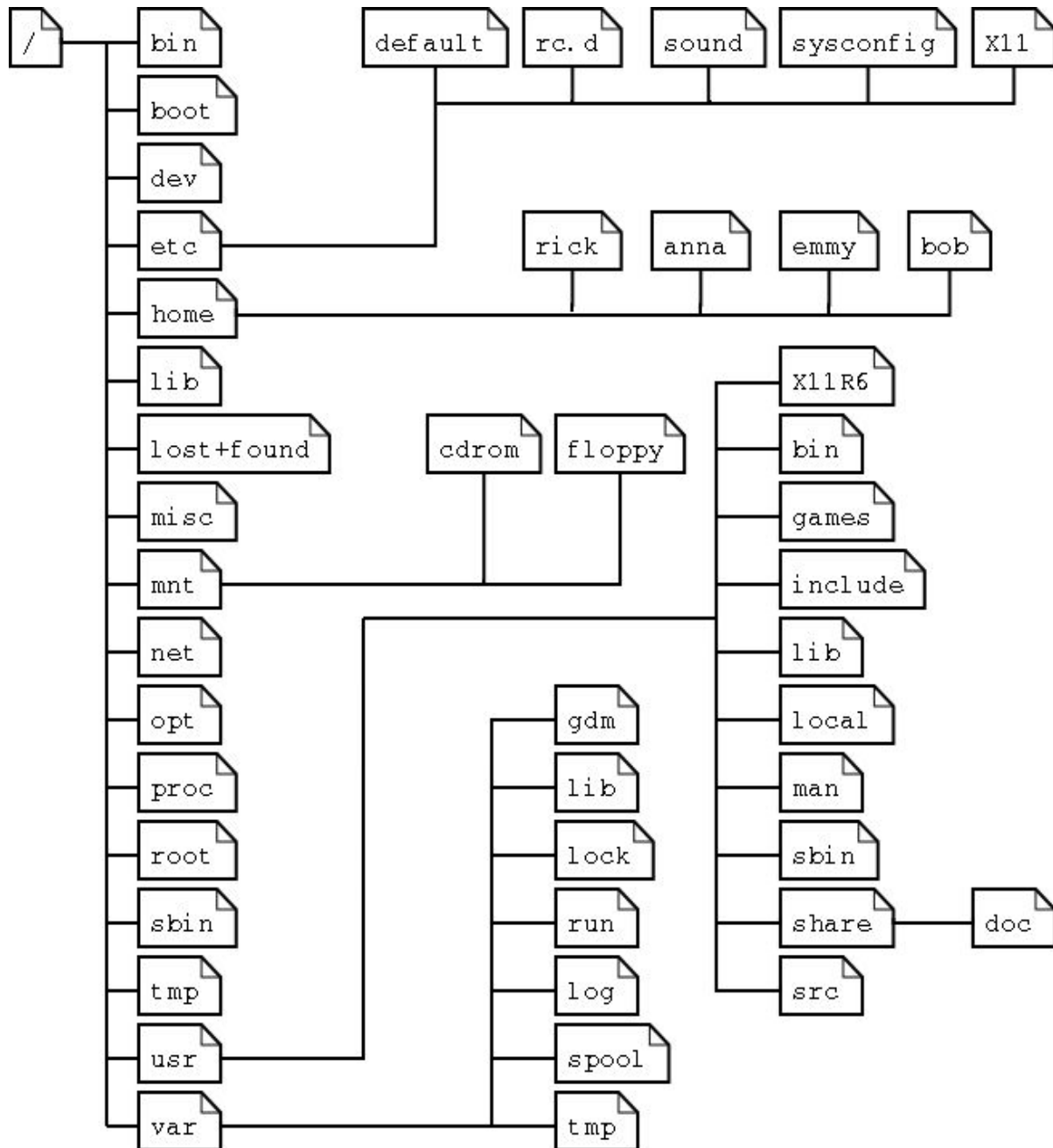
- The tilde (~) symbol stands for your home directory. If you are user, then the tilde (~) stands for /home/user
- **pwd**: The **pwd** command will allow you to know in which directory you're located (**pwd** stands for "print working directory"). Example: "**pwd**" in the Desktop directory will show "~/Desktop". Note that the GNOME Terminal also displays this information in the title bar of its window. A useful mnemonic is "present working directory."
- **ls**: The **ls** command will show you ('list') the files in your current directory. Used with certain options, you can see sizes of files, when files were made, and permissions of files. Example: "**ls ~**" will show you the files that are in your home directory.
- **cd**: The **cd** command will allow you to change directories. When you open a terminal you will be in your home directory. To move around the file system you will use **cd**. Examples:
  - To navigate into the root directory, use "**cd /**"
  - To navigate to your home directory, use "**cd**" or "**cd ~**"
  - To navigate up one directory level, use "**cd ..**"
  - To navigate to the previous directory (or back), use "**cd -**"

- To navigate through multiple levels of directory at once, specify the full directory path that you want to go to. For example, use, "**cd /var/www**" to go directly to the /www subdirectory of /var/. As another example, "**cd ~/Desktop**" will move you to the Desktop subdirectory inside your home directory.
- **cp**: The **cp** command will make a copy of a file for you. Example: "**cp file foo**" will make an exact copy of "file" and name it "foo", but the file "file" will still be there. If you are copying a directory, you must use "**cp -r directory foo**" (copy recursively). (To understand what "recursively" means, think of it this way: to copy the directory and all its files and subdirectories and all their files and subdirectories of the subdirectories and all their files, and on and on, "recursively")
- **mv**: The **mv** command will move a file to a different location or will rename a file. Examples are as follows: "**mv file foo**" will rename the file "file" to "foo". "**mv foo ~/Desktop**" will move the file "foo" to your Desktop directory, but it will not rename it. You must specify a new file name to rename a file.
  - To save on typing, you can substitute '~' in place of the home directory.
  - Note that if you are using **mv** with **sudo** you can use the ~ shortcut, because the terminal expands the ~ to your home directory. However, when you open a root shell with **sudo -i** or **sudo -s**, ~ will refer to the root account's home directory, not your own.
- **rm**: Use this command to remove or delete a file in your directory.
- **rmdir**: The **rmdir** command will delete an empty directory. To delete a directory and all of its contents recursively, use **rm -r** instead.
- **mkdir**: The **mkdir** command will allow you to create directories. Example: "**mkdir music**" will create a directory called "music".
- **man**: The **man** command is used to show you the manual of other commands. Try "**man man**" to get the man page for **man** itself. See the "**Man & Getting Help**" section down the page for more information.
- **sudo**: The **sudo** command is used to perform file operations on files that the **Root User** would only be allowed to change. An example would be trying to move one of your documents that another user accidentally moved to / back to your **documents** directory. Normally, to move the file, you would type **mv /mydoc.odt ~/Documents/mydoc.odt**, but you are not allowed to modify files outside of your

home directory. To get around this, you would type **sudo mv /mydoc.odt ~/Documents/mydoc.odt**. This will successfully move the file back to its correct location, provided that you are not a **standard** user, who has less (administrative) ability than an **administrator**. Be aware, though, that by using the **sudo** command, you need to be **extra** careful. It is easier to **damage** your system by using the **sudo** command. For more information about the **sudo** command,

## 2. Linux commands for redirection, pipes, filters, job control, file ownership, file permissions, links and file system hierarchy.

### File System



# File Permissions

## Understanding and Using File Permissions

In Linux and Unix, everything is a file. Directories are files, files are files and devices are files. Devices are usually referred to as a node; however, they are still files. All of the files on a system have permissions that allow or prevent others from viewing, modifying or executing. If the file is of type Directory then it restricts different actions than files and device nodes. The super user "root" has the ability to access any file on the system. Each file has access restrictions with permissions, user restrictions with owner/group association. Permissions are referred to as bits.

To change or edit files that are owned by root, sudo must be used

If the owner read & execute bit are on, then the permissions are:

**-r-X-----**

**There are three types of access restrictions:**

Permission	Action	chmod option
read	(view)	r or 4
write	(edit)	w or 2
execute	(execute)	x or 1

**There are also three types of user restrictions:**

User	ls output
owner	-rwx-----
group	----rwx---
other	-----rwx

**Note:** The restriction type scope is not inheritable: the file owner will be unaffected by restrictions set for his group or everybody else.

## Folder/Directory Permissions

Directories have directory permissions. The directory permissions restrict different actions than with files or device nodes.

Permission	Action	chmod option
read	(view contents, i.e. ls command)	r or 4
write	(create or remove files from dir)	w or 2
execute	(cd into directory)	x or 1

- read restricts or allows viewing the directories contents, i.e. ls command
- write restricts or allows creating new files or deleting files in the directory. (Caution: write access for a directory allows deleting of files in the directory even if the user does not have write permissions for the file!)
- execute restricts or allows changing into the directory, i.e. cd command

Folders (directories) must have 'execute' permissions set (x or 1), or folders (directories) will NOT FUNCTION as folders (directories) and WILL DISAPPEAR from view in the file browser (Nautilus).

## Permissions in Action

```
user@host:/home/user$ ls -l /etc/hosts
-rw-r--r-- 1 root root 288 2005-11-13 19:24 /etc/hosts
user@host:/home/user$
```

Using the example above we have the file "/etc/hosts" which is owned by the user root and belongs to the root group.

What are the permissions from the above /etc/hosts ls output?

```
-rw-r--r--
```



**owner = Read & Write (rw-)**  
**group = Read (r--)**  
**other = Read (r--)**

## Changing Permissions

The command to use when modifying permissions is `chmod`. There are two ways to modify permissions, with numbers or with letters. Using letters is easier to understand for most people. When modifying permissions be careful not to create security problems. Some files are configured to have very restrictive permissions to prevent unauthorized access. For example, the `/etc/shadow` file (file that stores all local user passwords) does not have permissions for regular users to read or otherwise access.

```
user@host:/home/user# ls -l /etc/shadow
-rw-r----- 1 root shadow 869 2005-11-08 13:16 /etc/shadow
user@host:/home/user#
```

### Permissions:

**owner = Read & Write (rw-)**  
**group = Read (r--)**  
**other = None (---)**

### Ownership:

**owner = root**  
**group = shadow**

chmod with Letters

**Usage: `chmod {options} filename`**

Options	Definition
u	owner
g	group
o	other
a	all (same as ugo)

x	execute
w	write
r	read
+	add permission
-	remove permission
=	set permission

Here are a few examples of chmod usage with letters (try these out on your system).

First create some empty files:

```
user@host:/home/user$ touch file1 file2 file3 file4
```

```
user@host:/home/user$ ls -l
```

```
total 0
```

```
-rw-r--r-- 1 user user 0 Nov 19 20:13 file1
```

```
-rw-r--r-- 1 user user 0 Nov 19 20:13 file2
```

```
-rw-r--r-- 1 user user 0 Nov 19 20:13 file3
```

```
-rw-r--r-- 1 user user 0 Nov 19 20:13 file4
```

Add owner execute bit:

```
user@host:/home/user$ chmod u+x file1
```

```
user@host:/home/user$ ls -l file1
```

```
-rwxr--r-- 1 user user 0 Nov 19 20:13 file1
```

Add other write & execute bit:

```
user@host:/home/user$ chmod o+wx file2
```

```
user@host:/home/user$ ls -l file2
```

```
-rw-r--rwx 1 user user 0 Nov 19 20:13 file2
```

Remove group read bit:

```
user@host:/home/user$ chmod g-r file3
```

```
user@host:/home/user$ ls -l file3
```

```
-rw----r-- 1 user user 0 Nov 19 20:13 file3
```

Add read, write and execute to everyone:

```
user@host:/home/user$ chmod ugo+rwx file4
user@host:/home/user$ ls -l file4
-rwxrwxrwx 1 user user 0 Nov 19 20:13 file4
user@host:/home/user$
```

chmod with Numbers

**Usage: chmod {options} filename**

Options	Definition
#--	owner
-#-	group
--#	other
1	execute
2	write
4	read

Owner, Group and Other is represented by three numbers. To get the value for the options determine the type of access needed for the file then add.

For example if you want a file that has -rw-rw-rwx permissions you will use the following:

Owner	Group	Other
read & write	read & write	read, write & execute
4+2=6	4+2=6	4+2+1=7

```
user@host:/home/user$ chmod 667 filename
```

Another example if you want a file that has --w-r-x--x permissions you will use the following:

Owner	Group	Other
write	read & execute	execute
2	4+1=5	1

**user@host:/home/user\$ chmod 251 filename**

Here are a few examples of chmod usage with numbers (try these out on your system).

First create some empty files:

**user@host:/home/user\$ touch file1 file2 file3 file4**

**user@host:/home/user\$ ls -l**

**total 0**

**-rw-r--r-- 1 user user 0 Nov 19 20:13 file1**

**-rw-r--r-- 1 user user 0 Nov 19 20:13 file2**

**-rw-r--r-- 1 user user 0 Nov 19 20:13 file3**

**-rw-r--r-- 1 user user 0 Nov 19 20:13 file4**

Add owner execute bit:

**user@host:/home/user\$ chmod 744 file1**

**user@host:/home/user\$ ls -l file1**

**-rwxr--r-- 1 user user 0 Nov 19 20:13 file1**

Add other write & execute bit:

**user@host:/home/user\$ chmod 647 file2**

**user@host:/home/user\$ ls -l file2**

**-rw-r--rwx 1 user user 0 Nov 19 20:13 file2**

Remove group read bit:

**user@host:/home/user\$ chmod 604 file3**

**user@host:/home/user\$ ls -l file3**

**-rw----r-- 1 user user 0 Nov 19 20:13 file3**

Add read, write and execute to everyone:

**user@host:/home/user\$ chmod 777 file4**

**user@host:/home/user\$ ls -l file4**

```
-rwxrwxrwx 1 user user 0 Nov 19 20:13 file4
user@host:/home/user$
```

### chmod with sudo

Changing permissions on files that you do not have ownership of: (Note that changing permissions the wrong way on the wrong files can quickly mess up your system a great deal! Please be careful when using sudo!)

```
user@host:/home/user$ ls -l /usr/local/bin/somefile
-rw-r--r-- 1 root root 550 2005-11-13 19:45 /usr/local/bin/somefile
user@host:/home/user$
```

```
user@host:/home/user$ sudo chmod o+x /usr/local/bin/somefile
```

```
user@host:/home/user$ ls -l /usr/local/bin/somefile
-rw-r--r-x 1 root root 550 2005-11-13 19:45 /usr/local/bin/somefile
user@host:/home/user$
```

## Recursive Permission Changes

To change the permissions of multiple files and directories with one command. Please note the warning in the chmod with sudo section and the Warning with Recursive chmod section.

### Recursive chmod with -R and sudo

To change all the permissions of each file and folder under a specified directory at once, use sudo chmod with -R

```
user@host:/home/user$ sudo chmod 777 -R /path/to/someDirectory
user@host:/home/user$ ls -l
total 3
-rwxrwxrwx 1 user user 0 Nov 19 20:13 file1
drwxrwxrwx 2 user user 4096 Nov 19 20:13 folder
-rwxrwxrwx 1 user user 0 Nov 19 20:13 file2
```

### Recursive chmod using find, pipemill, and sudo

To assign reasonably secure permissions to files and folders/directories, it's common to give files a permission of 644, and directories a 755 permission, since chmod -R assigns to both. Use sudo, the find command, and a pipemill to chmod as in the following examples.

To change permission of only files under a specified directory.

```
user@host:/home/user$ sudo find /path/to/someDirectory -type f -print0 | xargs -0 sudo chmod 644
```

```
user@host:/home/user$ ls -l
```

```
total 3
```

```
-rw-r--r-- 1 user user 0 Nov 19 20:13 file1
```

```
drwxrwxrwx 2 user user 4096 Nov 19 20:13 folder
```

```
-rw-r--r-- 1 user user 0 Nov 19 20:13 file2
```

To change permission of only directories under a specified directory (including that directory):

```
user@host:/home/user$ sudo find /path/to/someDirectory -type d -print0 | xargs -0 sudo chmod 755
```

```
user@host:/home/user$ ls -l
```

```
total 3
```

```
-rw-r--r-- 1 user user 0 Nov 19 20:13 file1
```

```
drwxr-xr-x 2 user user 4096 Nov 19 20:13 folder
```

```
-rw-r--r-- 1 user user 0 Nov 19 20:13 file2
```

## Warning with Recursive chmod

**WARNING:** Although it's been said, it's worth mentioning in context of a gotcha typo. Please note, Recursively deleting or chown-ing files are extremely dangerous. You will not be the first, nor the last, person to add one too many spaces into the command. This example will hose your system:

```
user@host:/home/user$ sudo chmod -R / home/john/Desktop/tempfiles
```

Note the space between the first / and home.

You have been warned.

## Changing the File Owner and Group

A file's owner can be changed using the chown command. For example, to change the foobar file's owner to tux:

```
user@host:/home/user$ sudo chown tux foobar
```

To change the foobar file's group to penguins, you could use either chgrp or chown with special syntax:

```
user@host:/home/user$ sudo chgrp penguins foobar
```

```
user@host:/home/user$ sudo chown :penguins foobar
```

Finally, to change the foobar file's owner to tux and the group to penguins with a single command, the syntax would be:

```
user@host:/home/user$ sudo chown tux:penguins foobar
```

## User Management

User management is a critical part of maintaining a secure system. Ineffective user and privilege management often lead many systems into being compromised. Therefore, it is important that you understand how you can protect your server through simple and effective user account management techniques.

### Where is root?

Ubuntu developers made a conscientious decision to disable the administrative root account by default in all Ubuntu installations. This does not mean that the root account has been deleted or that it may not be accessed. It merely has been given a password which matches no possible encrypted value, therefore may not log in directly by itself.

Instead, users are encouraged to make use of a tool by the name of *sudo* to carry out system administrative duties. *Sudo* allows an authorized user to temporarily elevate their privileges using their own password instead of having to know the password belonging to the root account. This simple yet effective methodology provides accountability for all user actions, and gives the administrator granular control over which actions a user can perform with said privileges.

**If for some reason you wish to enable the root account, simply give it a password:**

```
sudo passwd
```

**Sudo will prompt you for your password, and then ask you to supply a new password for root as shown below:**

```
[sudo] password for username: (enter your own password)
```

```
Enter new UNIX password: (enter a new password for root)
```

Retype new UNIX password: **(repeat new password for root)**

passwd: **password updated successfully**

**To disable the root account, use the following passwd syntax:**

```
sudo passwd -l root
```

**You should read more on *Sudo* by checking out it's man page:**

```
man sudo
```

By default, the initial user created by the Ubuntu installer is a member of the group "admin" which is added to the file `/etc/sudoers` as an authorized sudo user. If you wish to give any other account full root access through *sudo*, simply add them to the admin group.

## Adding and Deleting Users

The process for managing local users and groups is straight forward and differs very little from most other GNU/Linux operating systems. Ubuntu and other Debian based distributions, encourage the use of the "adduser" package for account management.

- To add a user account, use the following syntax, and follow the prompts to give the account a password and identifiable characteristics such as a full name, phone number, etc.

```
sudo adduser username
```

- To delete a user account and its primary group, use the following syntax:

```
sudo deluser username
```

Deleting an account does not remove their respective home folder. It is up to you whether or not you wish to delete the folder manually or keep it according to your desired retention policies.

Remember, any user added later on with the same UID/GID as the previous owner will now have access to this folder if you have not taken the necessary precautions.



You may want to change these UID/GID values to something more appropriate, such as the root account, and perhaps even relocate the folder to avoid future conflicts:

```
sudo chown -R root:root /home/username/  
sudo mkdir /home/archived_users/  
sudo mv /home/username /home/archived_users/
```

- To temporarily lock or unlock a user account, use the following syntax, respectively:

```
sudo passwd -l username  
sudo passwd -u username
```

- To add or delete a personalized group, use the following syntax, respectively:

```
sudo addgroup groupname  
sudo delgroup groupname
```

- To add a user to a group, use the following syntax:

```
sudo adduser username groupname
```

## How To Grant a User Sudo Privileges

If your new user should have the ability to execute commands with root (administrative) privileges, you will need to give the new user access to sudo. We can do this by using the visudo command, which opens the appropriate configuration file in your editor. This is the safest way to make these changes.

If you are currently signed in as the root user, type:

```
visudo
```

If you are signed in using a non-root user with sudo privileges, type:

`sudo visudo`

Search for the line that looks like this:

```
root    ALL=(ALL:ALL) ALL
```

Below this line, copy the format you see here, changing only the word "root" to reference the new user that you would like to give sudo privileges to:

```
root    ALL=(ALL:ALL) ALL
newuser ALL=(ALL:ALL) ALL
```

You should add a new line like this for each user that should be given full sudo privileges. When you are finished, you can save and close the file by hitting CTRL-X, followed by "Y", and then hit "ENTER" to confirm.

Now, your new user is able to execute commands with administrative privileges.

When signed in as the new user, you can execute commands as your regular user by typing commands as normal:

```
some_command
```

You can execute the same command with administrative privileges by typing sudo ahead of the command:

```
sudo some_command
```

You will be prompted to enter the password of the regular user account you are signed in as.

### 3. Shell Programming : Write shell script to show various system configuration

#### Shell Scripting

**NOTE:** The commands given in the scripting section are to be put into the text editor and not in the terminal unless instructed otherwise.

Bash is primarily a scripting language, so it would be a crime not to talk about scripting. Let's dive straight in with a bash script. More precisely the infamous "Hello World" script. You can create a bash script by opening your favorite text editor to edit your script and then saving it (typically the .sh file extension is used for your reference, but is not necessary. In our examples, we will be using the .sh extension).

```
#!/bin/bash
```

```
echo "Hello, World"
```

The first line of the script just defines which interpreter to use. NOTE: There is no leading whitespace before `#!/bin/bash`. That's it, simple as that. To run a bash script you first have to have the correct file permissions. We do this with [chmod](#) command in terminal (change mode) as follows:

```
chmod a+x /where/i/saved/it/hello_world.sh #Gives everyone execute permissions
# OR
chmod 700 /where/i/saved/it/hello_world.sh #Gives read,write,execute permissions to the Owner
```

This will give the file the appropriate permissions so that it can be executed. **Now open a terminal and run the script like this:**

```
/where/i/saved/it/hello_world.sh
```

Hopefully you should have seen it print Hello, World onto your screen. If so well done! That is your first Bash script.

TIP If you type:

```
pwd
```

You will see the directory that you are currently working in (pwd stands for 'print working directory'). If your current working directory is /where/i/saved/it/, then you can shorten the above command to:

```
prompt$ pwd
/where/i/saved/it
prompt$ ./hello_world.sh
```

### 3.1. logged user and his logname

```
x=$(logname)
echo "Currently logged user and his logname : $x"
```

### 3.2. Your current shell

```
echo $SHELL
```

### 3.3. Your home directory

```
echo $HOME
```

### 3.4. Your operating system type

```
x=$(arch)
echo "Your operating system type : $x"
```

### 3.5. Your current path setting

```
echo $PATH
```

3.6. Your current working directory

```
echo $pwd
```

3.7. Show Currently logged number of users

```
echo $users
```

## **4. Write shell script to show various system configuration like**

4.1. About your OS and version, release number, kernel version

```
x=$(lsb_release -a)
```

4.2. Show all available shells

```
cat /etc/shells
```

4.3. Show mouse settings

```
echo $(xinput --list --short)
```

4.4. Show computer CPU information like processor type, speed etc

```
echo $(sudo dmidecode -t 4)
```

Or

```
echo $(lscpu)
```

4.5. Show memory information

```
echo $(free -m)
```

4.6. Show hard disk information like size of hard-disk, cache memory, model etc

```
echo $(sudo dmidecode -t memory)
```

4.7. File system (Mounted)

```
echo $(sudo fdisk -l)
```

## 5. Shell script program for scientific calculator.

```
clear
sum=0
i="y"

echo " Enter one no."
read n1
echo "Enter second no."
read n2
while [ $i = "y" ]
do
echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"
echo "Enter your choice"
read ch
case $ch in
1)sum=`expr $n1 + $n2`
echo "Sum ="$sum;;
2)sum=`expr $n1 - $n2`
echo "Sub = "$sum;;
3)sum=`expr $n1 \* $n2`
echo "Mul = "$sum;;
4)sum=`expr $n1 / $n2`
echo "Div = "$sum;;
*)echo "Invalid choice";;
esac
echo
echo "Do u want to continue ?"
read i
if [ $i != "y" ]
then
exit
fi
done
```

## 7. Version Control System setup and usage using GIT.

### What is GIT

Git is a distributed revision control and source code management system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for Linux kernel development. Git is a free software distributed under the terms of the GNU General Public License version

### Installing GIT

The official website of Git has [detailed information about installing](#) on Linux, Mac, or Windows. In this case, we would be using Ubuntu 13.04 for demonstration purposes, where you install Git using apt-get.

```
sudo apt-get install git
```

### Initial Configuration

Let's create a directory inside which we will be working. Alternately, you could use Git to manage one of your existing projects, in which case you would not create the demo directory as below.

```
mkdir my_git_project  
cd my_git_project
```

The first step is to initialize Git in a directory. This is done using the command `init`, which creates a `.git` directory that contains all the Git-related information for your project.

```
donny@ubuntu:~$ mkdir my_git_project
donny@ubuntu:~$ cd my_git_project/
donny@ubuntu:~/my_git_project$ git init
Initialized empty Git repository in /home/donny/my_git_project/.git/
donny@ubuntu:~/my_git_project$
```

`git init`

Next, we need to configure our name and email. You can do it as follows, replacing the values with your own name and email.

```
git config --global user.name 'Shaumik'
git config --global user.email 'sd@gmail.com'
git config --global color.ui 'auto'
```

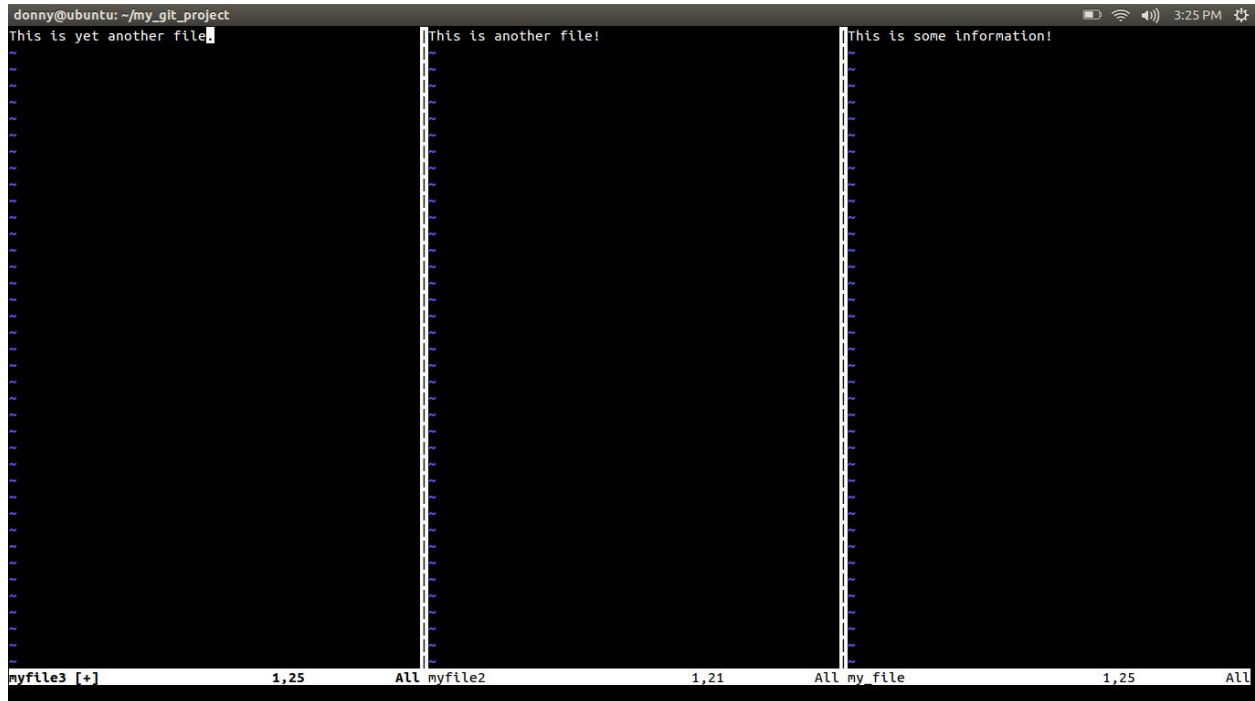
It is important to note that if you do not set your name and email, certain default values will be used. In our case, the username 'donny' and email 'donny@ubuntu' would be the default values.

Also, we set the UI color to `auto` so that the output of Git commands in the terminal are color coded. The reason we prefix `--global` to the command is to avoid typing these config commands the next time we start a Git project on our system.



## Staging Files for Commit

The next step is to create some files in the directory. You could use a text editor like [Vim](#). Note that if you are going to add Git to an already existing directory, you do not need to perform this step.



## Check the Status of Your Repository

Now that we have some files in our repository, let us see how Git treats them. To check the current status of your repository, we use the `git status` command.

`git status`

```
donny@ubuntu:~/my_git_project$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       my_file
#       myfile2
#       myfile3
nothing added to commit but untracked files present (use "git add" to track)
donny@ubuntu:~/my_git_project$
```

## Adding Files for Git to Track

At this point, we do not have any files for Git to track. We need to add files specifically to Git order to tell Git to track them. We add files using `add`.

```
git add my_file
```

Checking the status of the repository again shows us that one file has been added.

```
donny@ubuntu:~/my_git_project$ git add my_file
donny@ubuntu:~/my_git_project$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   my_file
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       myfile2
#       myfile3
donny@ubuntu:~/my_git_project$
```

To add multiple files, we use the following (note that we have added another file for demonstration purposes.)

```
git add myfile2 myfile3
```

You could use `git add` recursively, but be careful with that command. There are certain files (like compiled files) that are usually kept out of the Git repository. If you use `add` recursively, it would add all such files, if they are present in your repository.

## Removing Files

Let's say you have added files to Git that you do not want it to track. In such a situation, you tell Git to stop tracking them. Yet, running a simple `git rm` will not only remove it from Git, **but will also remove it from your local file system** as well! To tell Git to stop tracking a file, but still keep it on your local system, run the following command:

```
git rm --cached [file_name]
```

## Committing Changes

Once you have staged your files, you can **commit** them into Git. Imagine a commit as a snapshot in time where you can return back to access your repository at that stage. You associate a commit message with every commit, which you can provide with the `-m` prefix.

```
git commit -m "My first commit"
```

```
donny@ubuntu:~/my_git_project$ git commit -m "My first commit"
[master (root-commit) 8dd76fc] My first commit
 3 files changed, 3 insertions(+)
 create mode 100644 my_file
 create mode 100644 myfile2
 create mode 100644 myfile3
donny@ubuntu:~/my_git_project$
```

Provide a useful commit message because it helps you in identifying what you changed in that commit. Avoid overly general messages like “Fixed bugs”. If you have an issue tracker, you could provide messages like “Fixed bug #234”. It’s good practice to prefix your branch name or feature name to your commit message. For instance, “Asset management – Added feature to generate PDFs of assets” is a meaningful message.

Git identifies commits by attaching a long hexadecimal number to every commit. Usually, you do not need to copy the whole string, and the first 5-6 characters are enough to identify your commit. In the screenshot, notice that 8dd76fc identifies our first commit.

## Further Commits

Let’s now change a few files after our first commit. After changing them, we notice through `git status` that Git notices the change in the files that it is tracking.

```
donny@ubuntu:~/my_git_project$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   my_file
#       modified:   myfile2
#
no changes added to commit (use "git add" and/or "git commit -a")
donny@ubuntu:~/my_git_project$
```

You can check the changes to the tracked files from the last commit by running `git diff`. If you want to have a look at the changes to a particular file, you can run `git diff <file>`.

```
donny@ubuntu:~/my_git_project$ git diff
diff --git a/my_file b/my_file
index 362eab3..0a0bd57 100644
--- a/my_file
+++ b/my_file
@@ -1,3 @@
 This is some information!
+
+I am changing the content of this file.
diff --git a/myfile2 b/myfile2
index d4a2d15..ec4dcc2 100644
--- a/myfile2
+++ b/myfile2
@@ -1,1 @@
-This is another file!
+This is another file! Changing this file too.
donny@ubuntu:~/my_git_project$
```

You need to add these files again to stage the changes in tracked files for the next commit. You can add all the tracked files by running:

```
git add -u
```

You could avoid this command by prefixing `-a` to `git commit`, which adds all changes to tracked files for a commit. This process, however, is very dangerous as it can be damaging. For instance, let's say you opened a file and changed it by mistake. If you

selectively stage them, you would notice changes in each file. But if you add `-a` to your commit, all files would be committed and you would fail to notice possible errors.

Once you have staged your files, you can proceed to a commit. I mentioned that a message can be associated with every commit, which we entered by using `-m`.

However, it is possible for you to provide multi-line messages by using the command `git commit`, which opens up an interactive format for you to write!

`git commit`

```
GNU nano 2.2.6      File: .git/COMMIT_EDITMSG      Modified
- Changed two files
- This looks like a cooler interfact to write commit
messages
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   my_file
#       modified:   myfile2
#
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```



## Managing of Your Project

To check the history of your project, you can run the following command.

git log

```
donny@ubuntu:~/my_git_project$ git log
commit f934591cd1c04e4009dfa76a9684dda73cb30260
Author: Shaumik <sdaityari@gmail.com>
Date:   Tue May 6 15:31:00 2014 +0530

    - Changed two files
    - This looks like a cooler interfact to write commit
      messages

commit 8dd76fc3c4ba77b2ee795aed942cf6e9eaf5204d
Author: Shaumik <sdaityari@gmail.com>
Date:   Tue May 6 15:28:03 2014 +0530

    My first commit
donny@ubuntu:~/my_git_project$
```

This shows you the entire history of the project — which is a list of all the commits and their information. The information about a commit contains the commit hash, author, time and commit message. There are many variations of `git log`, which you could explore once you understand the concept of a **branch** in Git. To view the details of a particular commit and the files that were changed, run the following command:

git show <hash>

Where <hash> is the hex number associated with the commit. As this tutorial is for beginners, we will not cover how to get back to the state of a particular commit in time or how to manage branches.

```
donny@ubuntu:~/my_git_project$ git show 8dd76fc3c4ba7
commit 8dd76fc3c4ba77b2ee795aed942cf6e9eaf5204d
Author: Shaumik <sdaityari@gmail.com>
Date:   Tue May 6 15:28:03 2014 +0530

    My first commit

diff --git a/my_file b/my_file
new file mode 100644
index 00000000..362eab3
--- /dev/null
+++ b/my_file
@@ -0,0 +1 @@
+This is some information!
diff --git a/myfile2 b/myfile2
new file mode 100644
index 00000000..d4a2d15
--- /dev/null
+++ b/myfile2
@@ -0,0 +1 @@
+This is another file!
diff --git a/myfile3 b/myfile3
new file mode 100644
index 00000000..0eb26d0
--- /dev/null
+++ b/myfile3
@@ -0,0 +1 @@
+This is yet another file.
donny@ubuntu:~/my_git_project$
```



## Putting Your Code in the Cloud

Once you have learned how to manage your code on your system, the next step is to put it in the cloud. Since Git doesn't have a central server like Subversion, you need to add each source to collaborate with others. That is where the concept of remotes comes in. A remote refers to a remote version of your repository.

If you wish to put your code in the cloud, you could create a project on [GitHub](#), [GitLab](#), or [BitBucket](#) and push your existing code to the repository. In this case, the remote repository in the cloud would act as a remote to your repository. Conveniently, a remote to which you have write access is called the origin.

After you create a remote repository, you have the ability to add a remote origin and then push the code to the origin.

```
git remote add origin https://github.com/sdaityari/my_git_project.git
git push -u origin master
```

```
donny@ubuntu:~/my_git_project$ git remote add origin https://github.com/sdaityari/my_git_project.git
donny@ubuntu:~/my_git_project$ git push -u origin master
Username for 'https://github.com': sdaityari
Password for 'https://sdaityari@github.com':
Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 776 bytes, done.
Total 9 (delta 0), reused 0 (delta 0)
To https://github.com/sdaityari/my_git_project.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
donny@ubuntu:~/my_git_project$
```

## 9. Shell script to implement a script which kills every process which uses more than a specified value of memory or CPU and is run upon system start.

```
while [ 1 ];
do
echo
echo checking for run-away process ...

CPU_USAGE=$(uptime | cut -d", " -f4 | cut -d":" -f2 | cut -d" " -f2 | sed -e "s/\\.//g")
CPU_USAGE_THRESHOLD=800
PROCESS=$(ps aux r)
TOPPROCESS=$(ps -eo pid -eo pcpu -eo command | sort -k 2 -r | grep -v PID | head -n 1)

if [ $CPU_USAGE -gt $CPU_USAGE_THRESHOLD ] ; then
    kill -9 $(ps -eo pid | sort -k 1 -r | grep -v PID | head -n 1) #original
    kill -9 $(ps -eo pcpu | sort -k 1 -r | grep -v %CPU | head -n 1)
    kill -9 $TOPPROCESS
    echo system overloading!
    echo Top-most process killed $TOPPROCESS
    echo CPU USAGE is at $CPU_LOAD
else
    fi
    exit 0
    sleep 1;
done
```

# 11. Running PHP : simple applications like login forms after setting up a LAMP stack

## About LAMP

LAMP stack is a group of open source software used to get web servers up and running. The acronym stands for Linux, Apache, MySQL, and PHP. Since the virtual private server is already running Ubuntu, the linux part is taken care of. Here is how to install the rest.

## Set Up

The steps in this tutorial require the user to have root privileges on your VPS. You can see how to set that up in the Initial Server Setup in steps 3 and 4.

## Step 1: Install Apache

Apache is a free open source software which runs over 50% of the world's web servers.

To install apache, open terminal and type in these commands:

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

That's it. To check if Apache is installed, direct your browser to your server's IP address (eg. <http://12.34.56.789>). The page should display the words "It works!" like this.

## How to Find your Server's IP address

You can run the following command to reveal your server's IP address.

```
ifconfig eth0 | grep inet | awk '{ print $2 }'
```

## Step 2: Install MySQL

MySQL is a powerful database management system used for organizing and retrieving data

To install MySQL, open terminal and type in these commands:

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

During the installation, MySQL will ask you to set a root password. If you miss the chance to set the password while the program is installing, it is very easy to set the password later from within the MySQL shell.

Once you have installed MySQL, we should activate it with this command:

```
sudo mysql_install_db
```

Finish up by running the MySQL set up script:

```
sudo /usr/bin/mysql_secure_installation
```

The prompt will ask you for your current root password.

Type it in.

Enter current password for root (enter for none):

OK, successfully used password, moving on...

Then the prompt will ask you if you want to change the root password. Go ahead and choose N and move on to the next steps.

It's easiest just to say Yes to all the options. At the end, MySQL will reload and implement the new changes.

By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? [Y/n] y

... Success!

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y

... Success!

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

Remove test database and access to it? [Y/n] y

- Dropping test database...  
... Success!  
- Removing privileges on test database...  
... Success!

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Reload privilege tables now? [Y/n] y  
... Success!

Cleaning up...

Once you're done with that you can finish up by installing PHP.

## Step 3: Install PHP

PHP is an open source web scripting language that is widely use to build dynamic webpages.

To install PHP, open terminal and type in this command.

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

After you answer yes to the prompt twice, PHP will install itself.

It may also be useful to add php to the directory index, to serve the relevant php index files:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

Add index.php to the beginning of index files. The page should now look like this:

```
<IfModule mod_dir.c>
```

```
    DirectoryIndex index.php index.html index.cgi index.pl index.php index.xhtml index.htm
```

```
</IfModule>
```

## PHP Modules

PHP also has a variety of useful libraries and modules that you can add onto your virtual server. You can see the libraries that are available.

```
apt-cache search php5-
```

Terminal will then display the list of possible modules. The beginning looks like this:

```
php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
```

```
php5-cli - command-line interpreter for the php5 scripting language
```

```
php5-common - Common files for packages built from the php5 source
```

```
php5-curl - CURL module for php5
```

```
php5-dbg - Debug symbols for PHP5
```

```
php5-dev - Files for PHP5 module development
```

```
php5-gd - GD module for php5
```

```
php5-gmp - GMP module for php5
```

```
php5-ldap - LDAP module for php5
```

```
php5-mysql - MySQL module for php5
```

```
php5-odbc - ODBC module for php5
```

```
php5-pgsql - PostgreSQL module for php5
```

```
php5-pspell - pspell module for php5
```

```
php5-recode - recode module for php5
```

php5-snmp - SNMP module for php5  
php5-sqlite - SQLite module for php5  
php5-tidy - tidy module for php5  
php5-xmlrpc - XML-RPC module for php5  
php5-xsl - XSL module for php5  
php5-adodb - Extension optimising the ADOdb database abstraction library  
php5-auth-pam - A PHP5 extension for PAM authentication  
[...]

Once you decide to install the module, type:

```
sudo apt-get install name of the module
```

You can install multiple libraries at once by separating the name of each module with a space.

Congratulations! You now have LAMP stack on your droplet!

## Step 4: RESULTS — See PHP on your Server

Although LAMP is installed, we can still take a look and see the components online by creating a quick php info page

To set this up, first create a new file:

```
sudo nano /var/www/info.php
```

Add in the following line:



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

Then Save and Exit.

Restart apache so that all of the changes take effect:

```
sudo service apache2 restart
```

This tutorial demonstrates how to create a login page with MySQL Data base. Before enter into the code part, You would need special privileges to create or to delete a MySQL database. So assuming you have access to root user, you can create any database using mysql mysqladmin binary.

## Config.php

Config.php file is having information about MySQL Data base configuration.

```
<?php
define('DB_SERVER', 'localhost:3036');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', 'rootpassword');
define('DB_DATABASE', 'database');
$db = mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATABASE);
?>
```

## Login.php

Login PHP is having information about php script and HTML script to do login.

```
<?php
include("config.php");
session_start();

if($_SERVER["REQUEST_METHOD"] == "POST") {
    // username and password sent from form

    $myusername = mysqli_real_escape_string($db,$_POST['username']);
    $mypassword = mysqli_real_escape_string($db,$_POST['password']);

    $sql = "SELECT id FROM admin WHERE username = '$myusername' and passcode = '$mypassword'";
    $result = mysqli_query($db,$sql);
    $row = mysqli_fetch_array($result,MYSQLI_ASSOC);
    $active = $row['active'];

    $count = mysqli_num_rows($result);

    // If result matched $myusername and $mypassword, table row must be 1 row

    if($count == 1) {
        session_register("myusername");
        $_SESSION["login_user"] = $myusername;

        header("location: welcome.php");
    }else {
        $error = "Your Login Name or Password is invalid";
    }
}
?>
<html>

<head>
<title>Login Page</title>

<style type = "text/css">
    body {
        font-family:Arial, Helvetica, sans-serif;
        font-size:14px;
    }

    label {
        font-weight:bold;
        width:100px;
        font-size:14px;
    }
```

```

        .box {
            border:#666666 solid 1px;
        }
    </style>

</head>

<body bgcolor = "#FFFFFF">

    <div align = "center">
        <div style = "width:300px; border: solid 1px #333333; " align = "left">
            <div style = "background-color:#333333; color:#FFFFFF; padding:3px;"><b>Login</b></div>

            <div style = "margin:30px">

                <form action = "" method = "post">
                    <label>UserName :</label><input type = "text" name = "username" class = "box"/><br /><br />
                    <label>Password :</label><input type = "password" name = "password" class = "box" /><br/><br />
                    <input type = "submit" value = " Submit "/><br />
                </form>

                <div style = "font-size:11px; color:#cc0000; margin-top:10px"><?php echo $error; ?></div>

            </div>

        </div>

    </div>

</body>
</html>

```

# welcome.php

After successful login, it will display welcome page.

```

<?php
    include('session.php');
?>
<html">

    <head>
        <title>Welcome </title>
    </head>

    <body>

```

```

    <h1>Welcome <?php echo $login_session; ?></h1>
    <h2><a href = "logout.php">Sign Out</a></h2>
</body>

```

```

</html>

```

## Logout page

Logout page is having information about how to logout from login session.

```

<?php
    session_start();

    if(session_destroy()) {
        header("Location: login.php");
    }
?>

```

## session.php

Session.php will verify the session, if there is no session it will redirect to login page.

```

<?php
    include('config.php');
    session_start();

    $user_check = $_SESSION['login_user'];

    $ses_sql = mysqli_query($db,"select username from admin where username = '$user_check' ");

    $row = mysqli_fetch_array($ses_sql,MYSQLI_ASSOC);

    $login_session = $row['username'];

    if(!isset($_SESSION['login_user'])){
        header("location:login.php");
    }
?>

```

## 12. Advanced linux commands curl, wget, ftp, ssh and grep

**curl** - Transfers data from or to a server, using one of the protocols: HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP or FILE. (To transfer multiple files use wget or FTP.)

## OPTIONS

Maximum time that you allow the whole operation to take. This is useful for preventing your batch jobs from hanging for hours due to slow networks or links going down. See also the `--connect-timeout` option.

Tag	Description
url	One or multiple URLs that will be fetched in sequence. Multiple URLs or parts of URLs can be specified by writing part sets within braces as in:  <code>http://site.{one,two,three}.com</code> or get sequences of alphanumeric series by using <code>[]</code> as in: <code>ftp://ftp.numericals.com/file[1-100].txt</code> <code>ftp://ftp.numericals.com/file[001-100].txt</code> (with leading zeros) <code>ftp://ftp.letters.com/file[a-z].txt</code>
-A "agent string"  <code>--user-agent "agent string"</code>	Specify the User-Agent string to send to the HTTP server. To encode blanks in the string, surround the string with single quote marks. This can also be set with <code>-H</code> , <code>--header</code> option. (HTTP)
-b name=data  <code>--cookie name=data</code>	Send the data to the HTTP server as a cookie. It is supposedly the data previously received from the server in a "Set-Cookie:" line. The data should be in the format "NAME1=VALUE1; NAME2=VALUE2".
-c filename  <code>--cookie-jar file name</code>	Save cookies to file after a completed operation. Curl writes all cookies previously read from a specified file as well as all cookies received from remote server(s). If no cookies are known, no file will be written. To write to stdout, set the file name to a single dash, "-".

--compressed	Request a compressed response using one of the algorithms curl supports (gzip), and save the uncompressed document. If this option is used and the server sends an unsupported encoding, curl will report an error.(HTTP)
-d @file -d "string" --data "string"	Send the specified data in an (HTTP) POST request, in the same way that a web browser does. This will pass the data using the content-type application/x-www-form-urlencoded. Compare to -F, --form.
-d, --data is the same as --data-ascii. To post data in pure binary, use --data-binary.	To URL-encode the value of a form field you may use --data-urlencode. Multiple data options will be merged together. Thus, using '-d name=daniel -d skill=lousy' would generate a post that looks like 'name=daniel&skill=lousy'. If the data starts with @, the rest should be a filename containing the data.
-F name=@file -F name=content --form name=content	Emulate a filled-in form in which a user has pressed the submit button. This will POST data using the Content-Type multipart/form-data according to RFC 2388. This enables uploading of binary files etc. If the data starts with @, the rest should be a filename. To just get the content part from a file, prefix the file name with the symbol <. The difference between @ and < is that @ makes a file get attached in the post as a file upload, while the < makes a text field and gets the contents for that text field from a file.
-k --insecure	This option explicitly allows curl to perform "insecure" SSL connections and transfers. All SSL connections are attempted in secure mode using the CA

	<p>certificate bundle installed by default. This makes all connections considered "insecure" fail unless -k, --insecure is used.(SSL).</p>
--limit-rate speed	<p>Specify the maximum transfer rate. This feature is useful if you have a limited pipe and you'd like your transfer not to use your entire bandwidth. The given speed is measured in bytes/second, unless a suffix is appended. Appending 'k' or 'K' will count the number as kilobytes/sec, 'm' or 'M' megabytes, while 'g' or 'G' makes it gigabytes/sec. Eg: 200K, 3m, 1G.</p>
-m seconds --max-time seconds	
-o file --output file	<p>Write output to file instead of stdout. If you are using {} or [] to fetch multiple documents, you can use '#' followed by a number in the file specifier. That variable will be replaced with the current string for the URL being fetched. Like in:</p> <pre>curl http://{one,two}.site.com -o "file_#1.txt"</pre> <p>or use several variables like:</p> <pre>curl http://{site,host}.host[1-5].com -o "#1_#2"</pre> <p>You may use this option as many times as the number of URLs you have. See also --create-dirs option to create the local directories dynamically. Specify '-' to force the output to stdout.</p>
-O	<p>Write output to a local file named like the remote file we get. (Only the file part of the remote file is used,</p>

--remote-name	the path is cut off.) The remote file name to use for saving is extracted from the given URL, nothing else. Consequentially, the file will be saved in the current working directory.
-s --silent	Silent or quiet mode. Don't show progress meter or error messages.
--trace-ascii file	Enable a full trace dump of all incoming and outgoing data, including descriptive information, to the given output file. Use "-" as filename to have the output sent to stdout. This option overrides previous uses of -v, --verbose or --trace-ascii. If this option is used several times, the last one will be used.
-T file --upload-file file	Transfer the specified local file to the remote URL. PUT If there is no file part in the specified URL, Curl will append the local file name. You must use a trailing / on the last directory to really prove to Curl that there is no file name or curl will think that the last directory name is the remote file name to use. Use the file name "-" to use stdin. You can specify one -T for each URL on the command line. Each -T + URL pair specifies what to upload and to where. curl also supports "globbing" of the -T argument, meaning that you can upload multiple files to a single URL like this:  curl -T "{file1,file2}" http://www.uploadtothissite.com  or even  curl -T "img[1-1000].png" ftp://ftp.picturemania.com/upload/



<p>-I</p> <p>--head</p>	<p>Fetch the HTTP-header only! (HTTP/FTP/FILE)</p> <p>HTTP-servers feature the command HEAD which this uses to get nothing but the header of a document.</p> <p>When used on an FTP or FILE file, curl displays the file size and last modification time only.</p>
<p>-u user:password</p> <p>--user user:password</p>	<p>The username and password to use for server authentication. Overrides -n, --netrc and --netrc-optional. If you just give the user name (without entering a colon) curl will prompt for a password. If you use an SSPI-enabled curl binary and do NTLM authentication, you can force curl to pick up the username and password from your environment by specifying a single colon with this option: "-u :". If this option is used several times, the last one will be used.</p>
<p>-w</p> <p>--write-out format</p>	<p>Define extra info to display on stdout after a completed and successful operation. The format is a string that may contain plain text mixed with any number of variables. The format string can be specified as "string", or to read from a file specify "@filename" to read the format from stdin use "@-". Various variables may be included in the format and will be substituted by curl (file size, ip address etc see man curl for details). variables are specified as %{variable_name} Output a newline using \n, a carriage return with \r and a tab space with \t.</p>
<p>-x host:port</p>	<p>Use the specified HTTP proxy. If the port number is not specified, it is assumed at port 1080.</p>

-x [protocol://][user:password@]proxyhost[:port]  --proxy [protocol://][user:password@]proxyhost[:port]	
-H "name: value" --header "name: value"	Add Header when getting a web page. You may specify any number of extra headers.
-H "name:" --header "name:"	Remove Header, remove an internal header.
-L --location	Follow redirects if the server reports that the requested page has moved (indicated with a Location: header and a 3XX response code)
-v --verbose	Make more verbose/talkative. Mostly useful for debugging.

## EXAMPLES

To send your password file to the server, where 'password' is the name of the form-field to which /etc/passwd will be the input:

```
$ curl -F password=@/etc/passwd www.mypasswords.com
```

To retrieve a web page and display in the terminal

```
$ curl http://www.tutorialspoint.com
```

To retrieve a web page and display header information

```
$ curl http://www.tutorialspoint.com -i
```

To retrieve a web page and save to a file.

```
$ curl http://www.tutorialspoint.com -O tutorialspoint.html
```

To retrieve a web page, or its redirected target

```
$ curl www.tutorialspoint.com/unix/  
$ curl www.tutorialspoint.com/unix/ --location
```

To limit the rate of data transfer to 1 Kilobytes/sec

```
$ curl http://www.tutorialspoint.com/unix/ --limit-rate 1k -o unix.html
```

To download via a proxy server

```
$ curl -x proxy.example.com:3128 http://www.tutorialspoint.com/unix/
```

## 12. Advanced linux commands curl, wget, ftp, ssh and grep

# Wget

## About wget

**wget** stands for "web get". It is a **command-line utility** which **downloads** files over **anetwork**.

**wget** is a free utility for **non-interactive** download of files from the **web**. It supports **HTTP**, **HTTPS**, and **FTP protocols**, as well as retrieval through **HTTP proxies**.

**wget** is non-interactive, meaning that it can work in the background, while the user is not logged on. This allows you to start a retrieval and disconnect from the system, letting **wget** finish the work. By contrast, most **web browsers** require constant user interaction, which make transferring a lot of data difficult.

**wget** can follow links in **HTML** and **XHTML** pages and create local versions of remote websites, fully recreating the **directory** structure of the original site. This is sometimes called "recursive downloading." While doing that, **wget** respects the **Robot** Exclusion Standard (**robots.txt**). **wget** can be instructed to convert the links in downloaded HTML files to the local files for offline viewing.

**wget** has been designed for robustness over slow or unstable network connections; if a download fails due to a network problem, it will keep retrying until the whole file has been retrieved. If the **server** supports regetting, it will instruct the server to continue the download from where it left off.

## Overview

The simplest way to use **wget** is to simply provide it with the location of a file to download over HTTP. For example, to download the file **http://website.com/files/file.zip**, this command:

```
wget http://website.com/files/file.zip
```

There are many options that allow you to use **wget** in different ways, for different purposes. These are outlined below.

## Installing wget

If your operating system is Ubuntu, or another Debian-based Linux distribution which uses APT for package management, you can install **wget** with **apt-get**:

```
sudo apt-get install wget
```

For other operating systems, see your package manager's documentation for information about how to locate the **wget** binary package and install it. Or, you can install it from source from the GNU website at <http://www.gnu.org/software/wget/>.

## wget syntax

### Basic Startup Options

- |                                       |   |
|---------------------------------------|---|
| <b>-V, --version</b>                  | Display the version of <b>wget</b> , and exit.  |
| <b>-h, --help</b>                     | Print a help message describing all of <b>wget</b> 's command-line options, and exit.   |
| <b>-b, --background</b>               | Go to background immediately after startup. If no output file is specified via the <b>-o</b> , output is redirected to <b>wget-log</b> .  |
| <b>-e command, --execute comm and</b> | Execute <i>command</i> as if it were a part of the file <b>.wgetrc</b> . A command thus invoked will be executed after the commands in <b>.wgetrc</b> , thus taking precedence over them. |

## FTP

File Transfer Protocol (FTP) is a network protocol used to copy a file from one computer to another over the Internet or LAN. FTP follows a client-server architecture which utilizes separate control and data connections between the ftp client and server. The default port for ftp is 21.

## ftp: Internet File Transfer Program

Use the following syntax to connect to transfer files to and from a remote network ftp site:

```
ftp ftp.example.com
ftp 1.2.3.4
ftp user@ftp.example.com
```

You must know ftp username and password for user-based password authentication or with anonymous user access use ftp as both username and password. In this example, you are connecting to ftp.freebsd.org with anonymous user access (open the terminal and type the following command):

```
$ ftp ftp.freebsd.org
```

Sample session:

```
Trying 87.51.34.132...
Connected to ftp.freebsd.org.
220 ftp.beastie.tdk.net FTP server (Version 6.00LS) ready.
Name (ftp.freebsd.org:vivek): ftp
331 Guest login ok, send your email address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
```

ftp>

When you enter your own loginname and password for the ftp.example.com server, it returns the prompt

ftp>

You need to type all commands in front of the ftp> prompt.

### **Task: List Current File**

Type the ls command at ftp> prompt:

ftp> **ls**

Sample outputs:

```
229 Entering Extended Passive Mode (|||60692|)
150 Opening ASCII mode data connection for '/bin/ls'.
total 10
drwxrwxr-x 2 0 5 512 Jul 19 2007 .snap
drwx----- 2 0 0 2048 Jul 19 2007 lost+found
drwxr-xr-x 3 1006 1006 512 Sep 21 2009 pub
drwxr-xr-x 3 1006 1006 512 Jun 5 2007 sup
drwxr-xr-x 4 1006 0 512 Sep 18 2009 www
226 Transfer complete.
ftp>
```

The above will list the names of the files in the current remote directory (the last name is file or dir name).

### **Task: Change Directory**

To change directory on the remote machine use cd command:

ftp> cd dirName

To change to pub directory, enter:

```
ftp> cd pub
```

Sample outputs:

```
250 CWD command successful.
```

### **Task: Download / Copy file**

To copy one file at a time from the remote ftp server to the local system use get command:

```
get fileName  
get fileName newFileName
```

In this example, download file resume.pdf in the current remote directory to (or on top of) a file with the same name, resume.pdf, in your current local directory:

```
ftp> get resume.pdf
```

Sample outputs:

```
local: resume.pdf remote: resume.pdf  
229 Entering Extended Passive Mode (|||55093|)  
150 Opening BINARY mode data connection for 'resume.pdf' (53077 bytes).  
100% |*****| 53077 12.58  
KiB/s 00:00 ETA  
226 Transfer complete.  
53077 bytes received in 00:04 (12.57 KiB/s)
```

In this example, copies file data.tar.gz in the current remote directory to (or on top of) a file named backup.tar.gz in your current local directory:

```
ftp> get data.tar.gz backup.tar.gz
```



### Change Local Directory

To change directory on your local system, enter:

```
ftp> lcd /path/to/new/dir
```

```
ftp> lcd /tmp
```

Sample outputs:

Local directory now: /tmp

Print local directory:

```
ftp> lpwd
```

Sample outputs:

/tmp

The lpwd command prints current download directory for local systems. However, to find out the pathname of the current directory on the remote ftp server, enter:

```
ftp> pwd
```

Sample outputs:

Remote directory: /pub/FreeBSD

### Task: Download Multiple Files

You need to use mget command as follows to copy multiple files from the remote ftp server to the local system. You may be prompted for a yes/no (Y/N) answer before transferring each file (you can disable prompt by passing the -i option to ftp client). To download all files, enter:

```
ftp> mget *
```

To download all perl files (ending with .pl extension), enter:

```
ftp> mget *.pl
```

### **Task: Turn On / Off Interactive Prompting**

The ftp command prompt sets interactive prompting; “on” which enables prompting so that you can verify of each step of the multiple commands, “off” allows the commands to act unimpeded:

```
ftp> prompt on
```

```
ftp> mput *.php
```

```
ftp> prompt off
```

```
ftp> mget *.py
```

### **Task: Delete File**

To delete a file in the current remote directory use delete command:

```
ftp> delete fileName
```

```
ftp> delete output.jpg
```

### **Task: Upload One File**

To copy one file at a time from the local systems to the remote ftp server, enter:

```
ftp> put fileName
```

In this example, upload logo.jpg, enter:

```
ftp> put logo.jpg
```

### **Task: Upload Multiple Files**

To copy multiple files from the local system to the remote ftp server use mput command. Again, you may be prompted for a yes/no (y/n) answer before transferring each file. In this example, upload all files from the current system:

```
ftp> mput *
```

```
ftp> mput *.pl
```

### **Task: Create a Directory**

To make a new directory, enter:

```
ftp> mkdir dirName
```

```
ftp> mkdir scripts
```

```
ftp> cd scripts
```

```
ftp> pwd
```

### **Task: Delete a Directory**

To remove or delete a directory, enter:

```
ftp> rmdir dirName
```

```
ftp> rmdir images
```

```
ftp> ls
```

### **Task: Set The Mode Of File Transfer**

To set the mode of file transfer to ASCII, enter:

```
ftp> ascii
```

Please note that ascii is the default and good for text files. To set the mode of file transfer to binary, enter:

```
ftp> binary
```

The binary mode is recommended for almost all sort of files including images, zip files and much more. The binary mode provides less chance of a transmission error.

### **Task: Connect To Another FTP Server**

To open a connection with another ftp server, enter:

```
ftp> open ftp.nixcraft.net.in
```

The above command opens a new FTP connection with ftp.nixcraft.net.in. You must provide a username and password for a ftp.nixcraft.net.in account. However, a username and password can be skipped for an anonymous FTP connection.

### **Task: Exit the FTP Session**

Type quit or bye, enter:

```
ftp> quit
```

OR

```
ftp> bye
```

Sample outputs:

```
221 Goodbye.
```

## **How Do I Find Out More Information About The FTP Commands?**

Type ? or help to get more information about the FTP commands:

ftp> ?

ftp> help

Sample outputs:

Commands may be abbreviated. Commands are:

!	delete	idle	mode	pmlsd	reset	
system						
\$	dir	image	modtime		preserve	restart
tenex						
account		disconnect	lcd	more	progress	rhel
throttle						
append		edit	less	mput	prompt	rmdir
	trace					
ascii	epsv4	lpage	mreget		proxy	rstatus
	type					
bell	exit	lpwd	msend		put	runique
	umask					
binary	features	ls	newer	pwd	send	unset
bye	fget	macdef	nlist		quit	sendport
usage						
case	form	mdelete	nmap		quote	set
user						
cd	ftp	mdir	ntrans	rate	site	
verbose						
cdup	gate	mget	open	rcvbuf	size	
xferbuf						
chmod		get	mkdir	page	recv	sndbuf
?						
close	glob	mls	passive		reget	status
cr	hash	mlsd	pdir	remopts		struct
debug	help	mlst	pls	rename		sunique

To get a short description about each command, enter:

```
ftp> help commandName
```

```
ftp> help chmod
```

Sample outputs:

```
chmod      change file permissions of remote file
```

```
ftp> help ls
```

Sample outputs:

```
ls          list contents of remote path
```

## FTP Through A Browser

If you do not want to type the commands, than use a browser such as Safari, Firefox and type the following:

```
ftp://ftpUserName@ftp.nixcraft.net.in
```

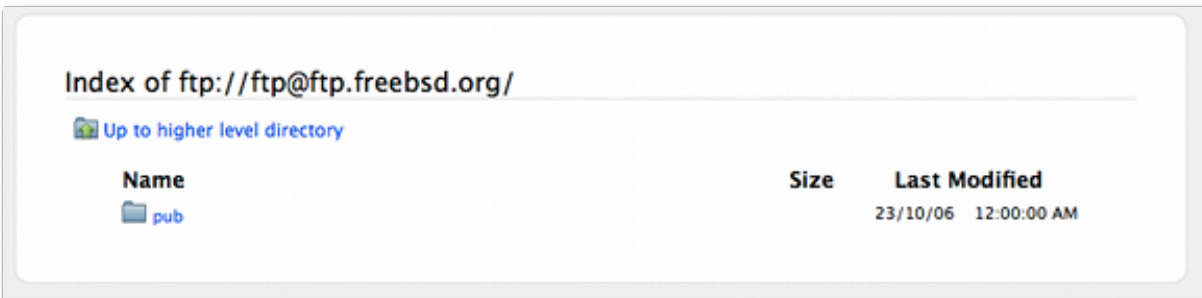
```
ftp://ftp.freebsd.org/
```

```
ftp://ftp@ftp.freebsd.org/
```

```
ftp://userName:Password@ftp.nixcraft.net.in/
```

```
ftp://ftp:ftp@ftp.freebsd.org/
```

Sample outputs:



# How To Use SSH to Connect to a Remote Server in Ubuntu

## What Is SSH?

One essential tool to master as a system administrator is SSH.

SSH, or *Secure Shell*, is a protocol used to securely log onto remote systems. It is the most common way to access remote Linux and Unix-like servers.

In this guide, we will discuss how to use SSH to connect to a remote system.

## Basic Syntax

The tool on Linux for connecting to a remote system using SSH is called, unsurprisingly, `ssh`.

The most basic form of the command is:

- `ssh remote_host`

The *remote\_host* in this example is the IP address or domain name that you are trying to connect to.

This command assumes that your username on the remote system is the same as your username on your local system.

If your username is different on the remote system, you can specify it by using this syntax:

- `ssh remote_username@remote_host`

Once you have connected to the server, you will probably be asked to verify your identity by providing a password.

Later, we will cover how to generate keys to use instead of passwords.

To exit back into your local session, simply type:

- `exit`

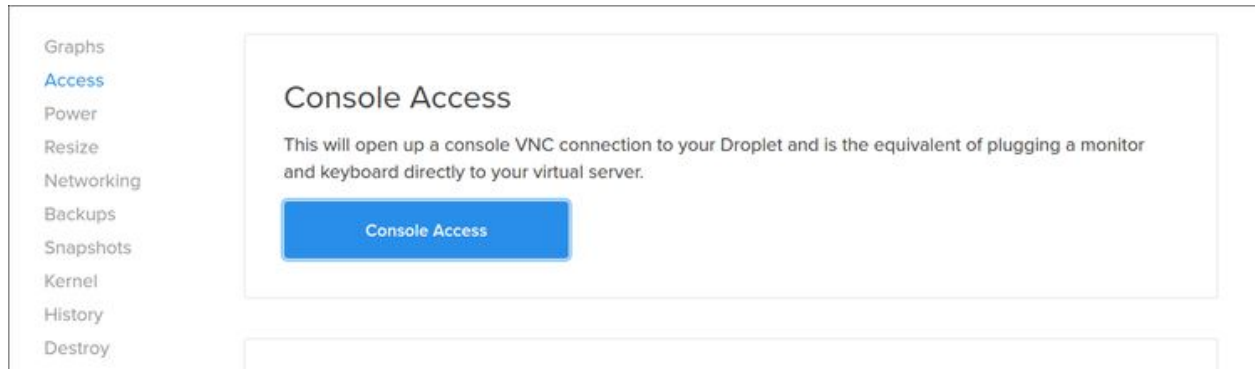
## How Does SSH Work?

SSH works by connecting a client program to an ssh server.

In the above commands, *ssh* is the client program. The *ssh server* is already running on the *remote\_host* that we specified.



In your Droplet, the `sshd` server should already be running. If this is not the case, click on the *Console Access* button from your Droplet page:



You will be presented with a login screen. Log in with your credentials.

The process needed to start an `ssh` server depends on the distribution of Linux that you are using.

On Ubuntu, you can start the `ssh` server on the Droplet by typing:

- `sudo service ssh start`

On Ubuntu 16.04 and Debian Jessie, you can use `systemctl`, the `systemd` command for managing services:

- `sudo systemctl start ssh`

That should start the `sshd` server and you can then log in remotely.

## How To Configure SSH

When you change the configuration of SSH, you are changing the settings of the sshd server.

In Ubuntu, the main sshd configuration file is located at `/etc/ssh/sshd_config`.

Back up the current version of this file before editing:

- `sudo cp /etc/ssh/sshd_config{,.bak}`

Open it with a text editor:

- `sudo nano /etc/ssh/sshd_config`

You will want to leave most of the options in this file alone. However, there are a few you may want to take a look at:

`/etc/ssh/sshd_config`

Port 22

The port declaration specifies which port the sshd server will listen on for connections. By default, this is 22. You should probably leave this setting alone, unless you have specific reasons to do otherwise. If you *do* change your port, we will show you how to connect to the new port later on.

`/etc/ssh/sshd_config`

HostKey /etc/ssh/ssh\_host\_rsa\_key  
HostKey /etc/ssh/ssh\_host\_dsa\_key  
HostKey /etc/ssh/ssh\_host\_ecdsa\_key

The host keys declarations specify where to look for global host keys. We will discuss what a host key is later.

```
/etc/ssh/sshd_config
```

```
SyslogFacility AUTH  
LogLevel INFO
```

These two items indicate the level of logging that should occur.

If you are having difficulties with SSH, increasing the amount of logging may be a good way to discover what the issue is.

```
/etc/ssh/sshd_config
```

```
LoginGraceTime 120  
PermitRootLogin yes  
StrictModes yes
```

These parameters specify some of the login information.

LoginGraceTime specifies how many seconds to keep the connection alive without successfully logging in.

It may be a good idea to set this time just a little bit higher than the amount of time it takes you to log in normally.

PermitRootLogin selects whether root is allowed to log in.

In most cases, this should be changed to "no" when you have created user account that has access to elevated privileges (through su or sudo) and can log in through ssh.

strictModes is a safety guard that will refuse a login attempt if the authentication files are readable by everyone.

This prevents login attempts when the configuration files are not secure.

```
/etc/ssh/sshd_config
```

```
X11Forwarding yes  
X11DisplayOffset 10
```

These parameters configure an ability called *X11 Forwarding*. This allows you to view a remote system's graphical user interface (GUI) on the local system.

This option must be enabled on the server and given with the SSH client during connection with the `-X` option.

If you changed any settings in `/etc/ssh/sshd_config`, make sure you restart your sshd server to implement your modifications:

- `sudo service ssh restart`

Or, on systemd systems such as Ubuntu 16.04 or Debian Jessie:

- `sudo systemctl restart ssh`

You should thoroughly test your changes to ensure that they operate in the way you expect.

It may be a good idea to have a few sessions active when you are making changes. This will allow you to revert the configuration if necessary.

If you run into problems, remember that you can log in through the *Console* link on your Droplet page.

## How To Log Into SSH with Keys

While it is helpful to be able to log in to a remote system using passwords, it's a much better idea to set up *key-based authentication*.

## How Does Key-based Authentication Work?

Key-based authentication works by creating a pair of keys: a *private key* and a *public key*.

The *private key* is located on the client machine and is secured and kept secret.

The *public key* can be given to anyone or placed on any server you wish to access.

When you attempt to connect using a key-pair, the server will use the public key to create a message for the client computer that can only be read with the private key.

The client computer then sends the appropriate response back to the server and the server will know that the client is legitimate.

This entire process is done in the background automatically after you set up keys.

## How To Create SSH Keys

SSH keys should be generated on the computer you wish to log in *from*. This is usually your local computer.

Enter the following into the command line:

- `ssh-keygen -t rsa`

Press enter to accept the defaults. Your keys will be created at `~/.ssh/id_rsa.pub` and `~/.ssh/id_rsa`.

Change into the `.ssh` directory by typing:

- `cd ~/.ssh`

Look at the permissions of the files:

- `ls -l`

Output

```
-rw-r--r-- 1 demo demo 807 Sep  9 22:15 authorized_keys
-rw----- 1 demo demo 1679 Sep  9 23:13 id_rsa
-rw-r--r-- 1 demo demo 396 Sep  9 23:13 id_rsa.pub
```

As you can see, the `id_rsa` file is readable and writable only to the owner. This is how it should be to keep it secret.

The `id_rsa.pub` file, however, can be shared and has permissions appropriate for this activity.

## How To Transfer Your Public Key to the Server

You can copy the public key to the remote server by issuing this command:

- `ssh-copy-id remote_host`

This will start an SSH session, which you will need to authenticate with your password.

After you enter your password, it will copy your public key to the server's authorized keys file, which will allow you to log in without the password next time.

## Client-Side Options

There are a number of optional flags that you can select when connecting through SSH.

Some of these may be necessary to match the settings in the remote host's `sshd` configuration.

For instance, you if you changed the port number in your `sshd` configuration, you will need to match that port on the client-side by typing:

- `ssh -p port_number remote_host`

If you only wish to execute a single command on a remote system, you can specify it after the host like so:

- `ssh remote_host command_to_run`

You will connect to the remote machine, authenticate, and the command will be executed.

As we said before, if X11 forwarding is enabled on both computers, you can access that functionality by typing:

- `ssh -X remote_host`

Providing you have the appropriate tools on your computer, GUI programs that you use on the remote system will now open their window on your local system.

## What Is grep?

**Grep** is a command-line tool that allows you to find a string in a file or stream. It can be used with a regular expression to be more flexible at finding strings.

This page gives an introduction to grep. For more information enter:

`man grep`

in a terminal.

## How To Use grep

In the simplest case grep can be invoked as follows:

`grep 'STRING' filename`

The above command searches the file for STRING and lists the lines that contain a match.

This is OK but it does not show the true power of grep. The above command only looks at one file. A cool example of using grep with multiple files would be to find all lines in all



files in a given directory that contain the name of a person. This can be easily accomplished as follows:

```
grep 'Nicolas Kassis' *
```

The above command searches all files in the current directory for the name and lists all lines that contain a match.

Notice the use of quotes in the above command. Quotes are not usually essential, but in this example they are essential because the name contains a space. Double quotes could also have been used in this example.

## Regular Expressions

grep can search for complicated patterns to find what you need. Here is a list of some of the special characters used to create a regular expression:

^	Denotes the beginning of a line
\$	Denotes the end of a line
.	Matches any single character
*	The preceding item in the regular expression will be matched zero or more times
[]	A bracket expression is a list of characters enclosed by [ and ]. It matches any single character in that list; if the first character of the list is the caret ^ then it matches any character not in the list
\<	Denotes the beginning of a word
\>	Denotes the end of a word

Here is an example of a regular expression search:

```
grep "\<[A-Za-z].*\>" file
```

This regular expression matches any "word" that begins with a letter (upper or lower case). For example, "words" that begin with a digit would not match. The grep command lists the lines that contain a match.

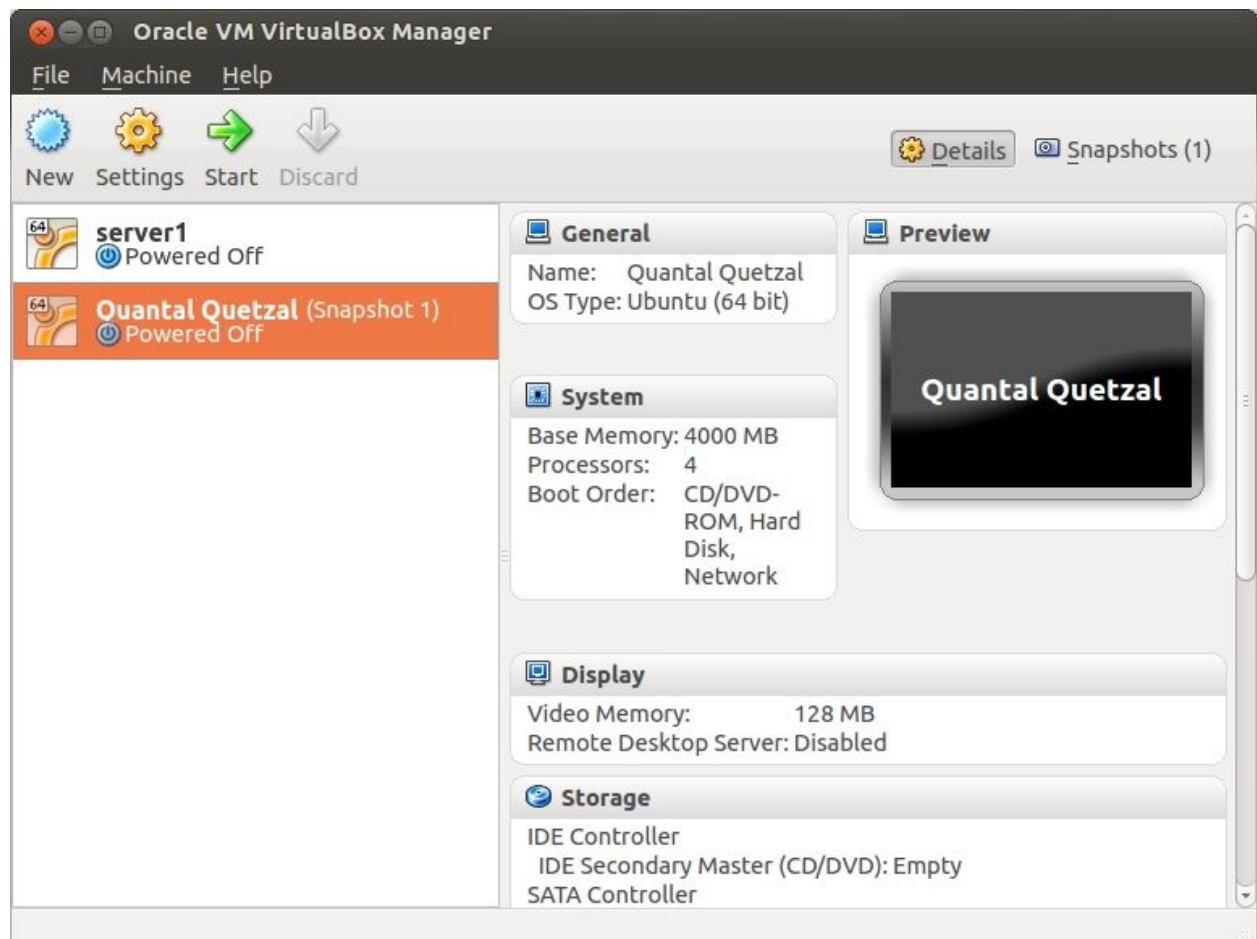
**15. Virtualisation environment (e.g., xen, kqemu, virtualbox or lguest) to test an applications, new kernels and isolate applications. It could also be used to expose students to other alternate OSs like \*BSD**

1. Open VirtualBox and select New . A new window will come out.
2. Choose your guest OS and architecture (32 vs. 64 bit, e.g select Ubuntu)
3. Set your Base Memory (RAM)
4. Click next until it show the vm storage size. Put how much space you need depending on your hardisk and finish the wizard by clicking the create button.
5. On VirtualBox main window, select START and pick your MEDIA SOURCE. In your case, select iso on your desktop.
6. Finish the installation as normal install.
7. Remove your installation .iso from the virtual optical disk drive before restarting the VM.
8. Install **Guest Additions**.

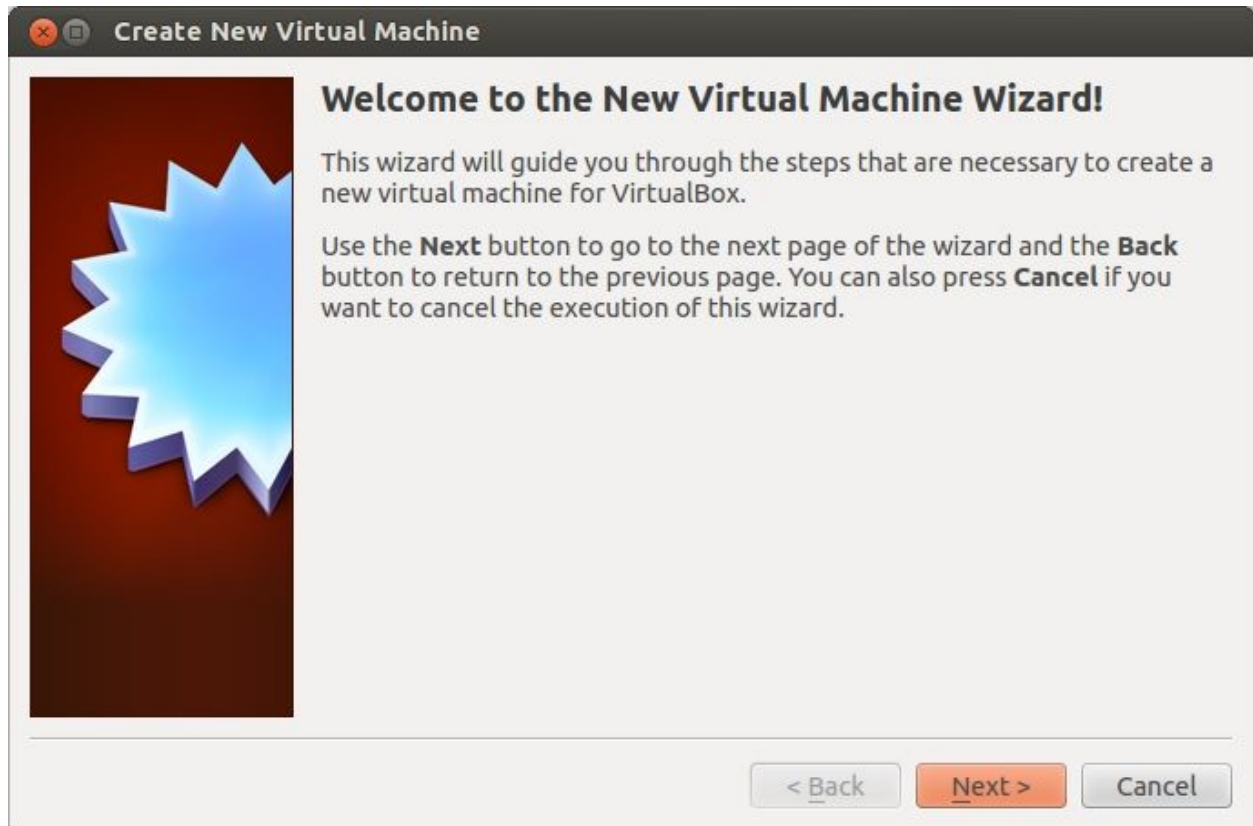
---

### **Follow this guide:**

Open Virtualbox and click at **New** button.

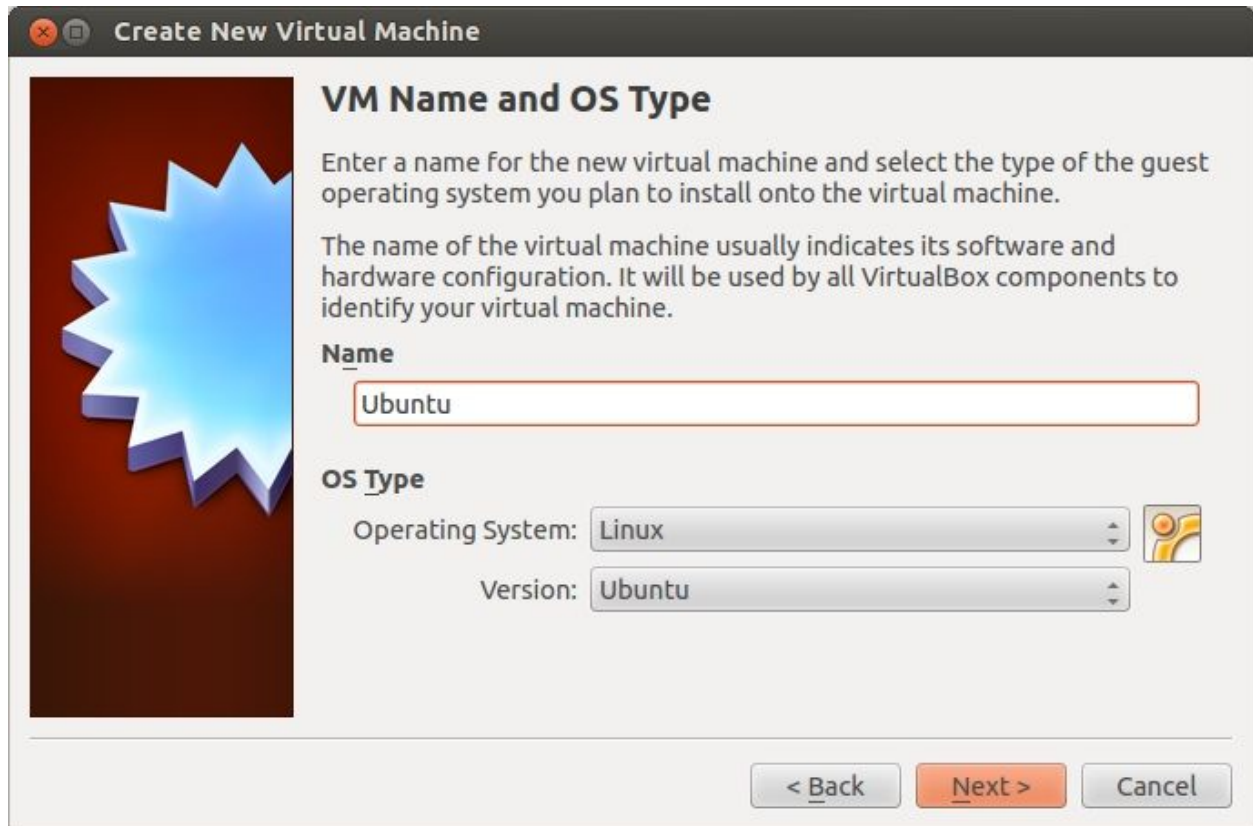


Setup Wizard will appear and click at **Next** button.



Enter your Virtual Machine name, and choose your guest OS and architecture (32- vs. 64-bit) from the dropdown menu and click **Next** button.

A 64-bit guest needs the **CPU virtualization technology (VT-x AMD/V)** to be enabled in BIOS.



The image shows a 'Create New Virtual Machine' dialog box. On the left is a decorative graphic of a blue starburst on a dark red background. The main area is titled 'VM Name and OS Type'. It contains instructions to enter a name and select an OS type. A text field contains 'Ubuntu'. Below it, 'OS Type' is shown with 'Operating System' set to 'Linux' and 'Version' set to 'Ubuntu'. At the bottom are '< Back', 'Next >', and 'Cancel' buttons.

### Create New Virtual Machine

#### VM Name and OS Type

Enter a name for the new virtual machine and select the type of the guest operating system you plan to install onto the virtual machine.

The name of the virtual machine usually indicates its software and hardware configuration. It will be used by all VirtualBox components to identify your virtual machine.

**Name**

Ubuntu

**OS Type**

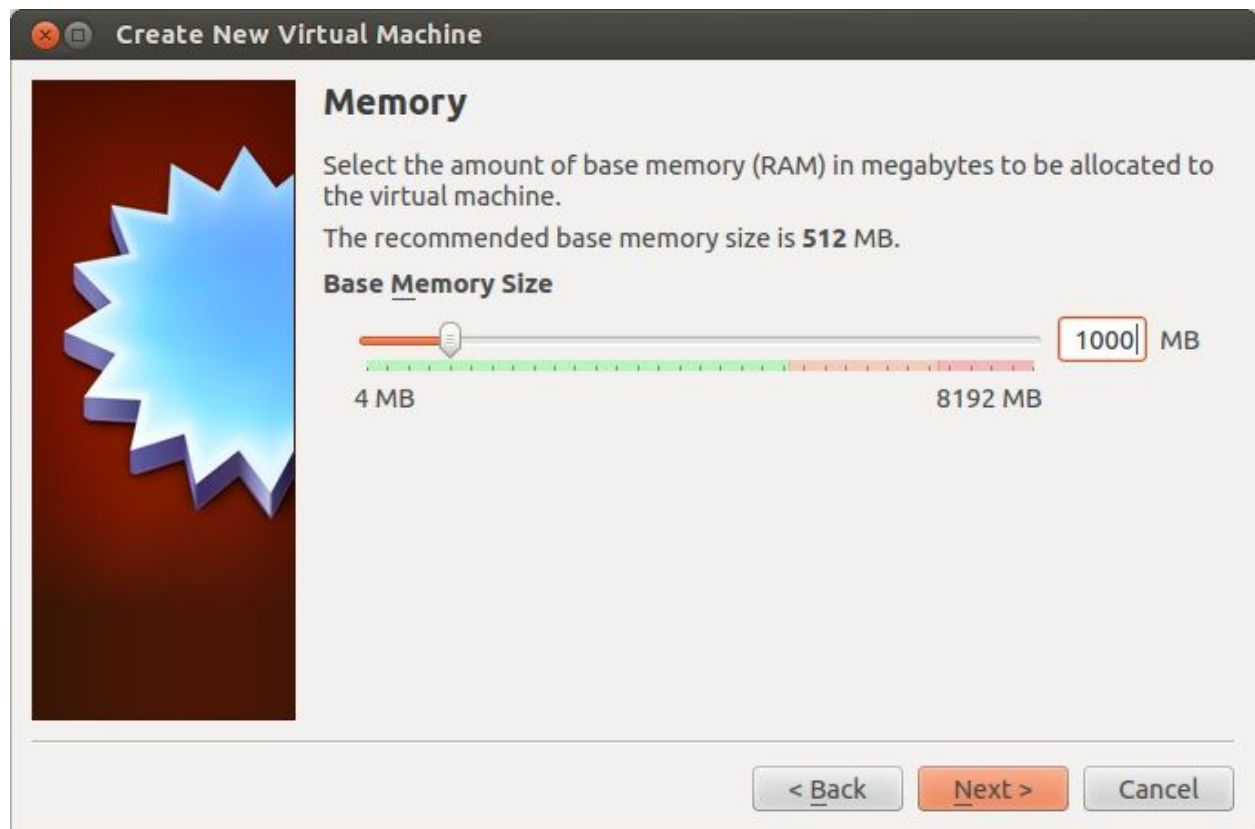
Operating System: Linux

Version: Ubuntu

< Back   Next >   Cancel

Enter memory (RAM) to reserve for your virtual machine and click **Next** button.

Leave enough memory to the host OS.



Tick at **Startup Disk** and **Create New Hard disk** and click at **Next** button.



Choose the type of file that you want to use for virtual disk and click **Next** button.



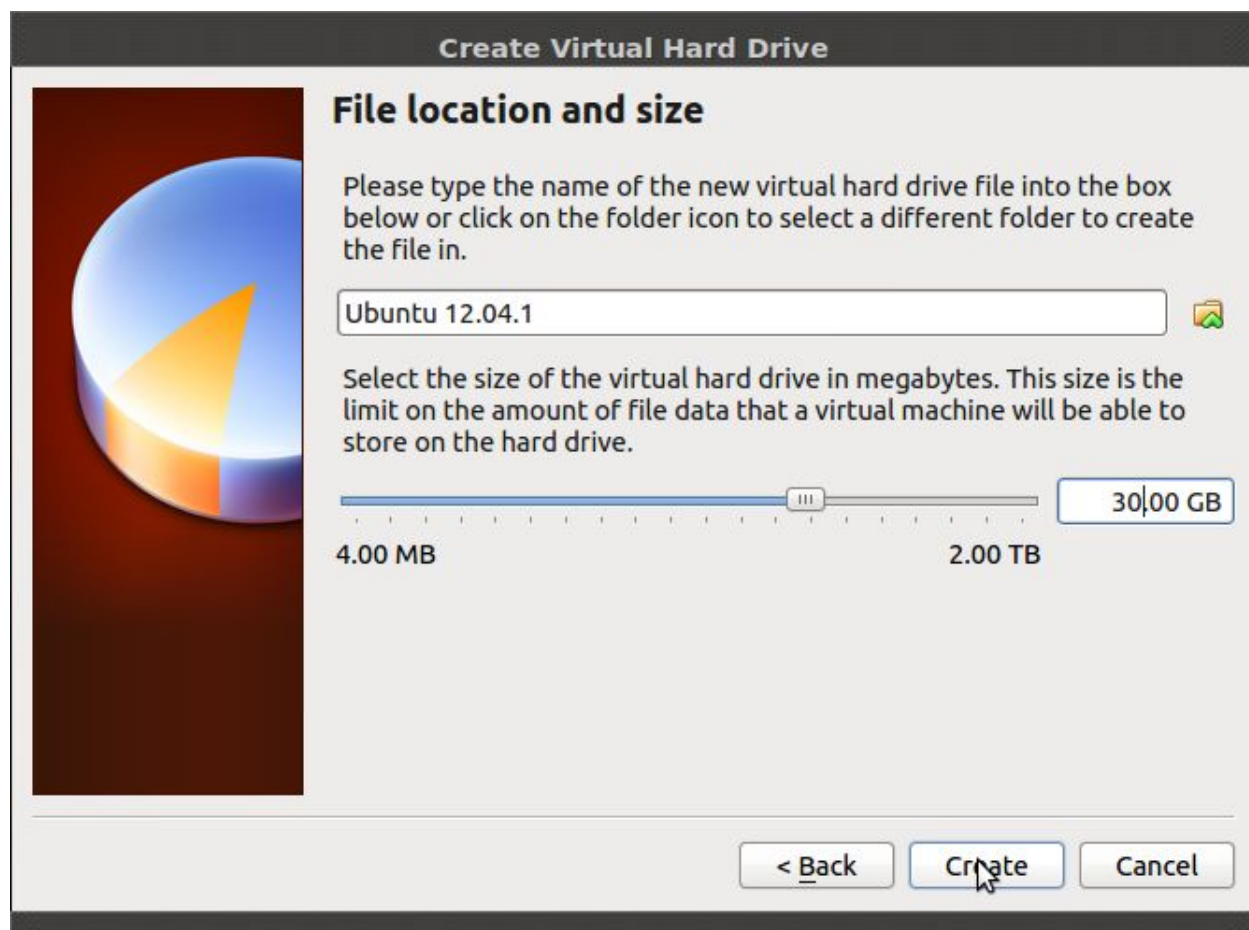
Choose your storage detail and click **Next** button.





Enter the size of your virtual disk (in MB) and click **Next** button.

A dynamically growing virtual disk will only use the amount of physical hard drive space it needs. It is better to be rather generous to avoid running out of guest hard drive space.



## Create Virtual Hard Drive

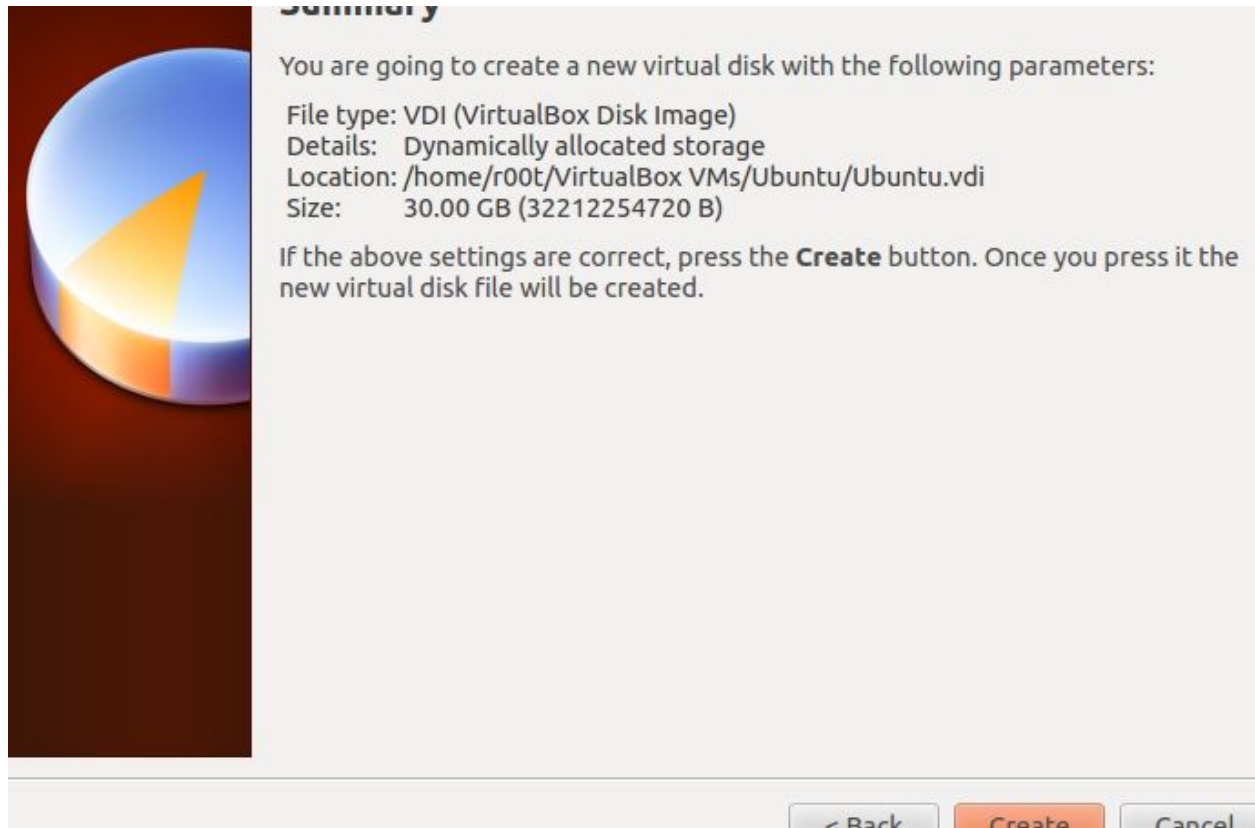
Please type the name of the new virtual hard drive file into the box below or click on the folder icon to select a different folder to create the file in.



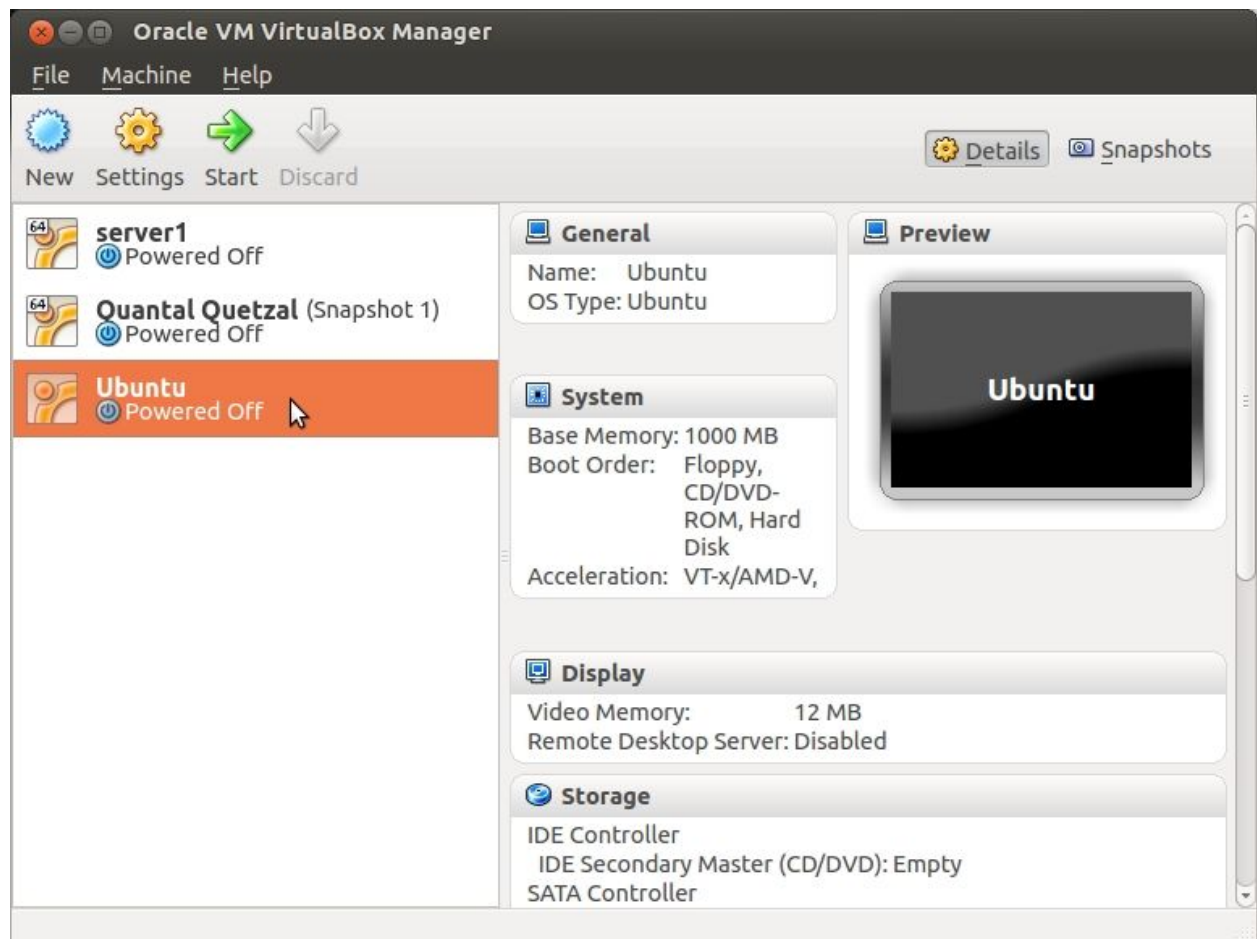
Select the size of the virtual hard drive in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard drive.

[< Back](#)

You will see the detail of your input here. Click **Create** button to continue.



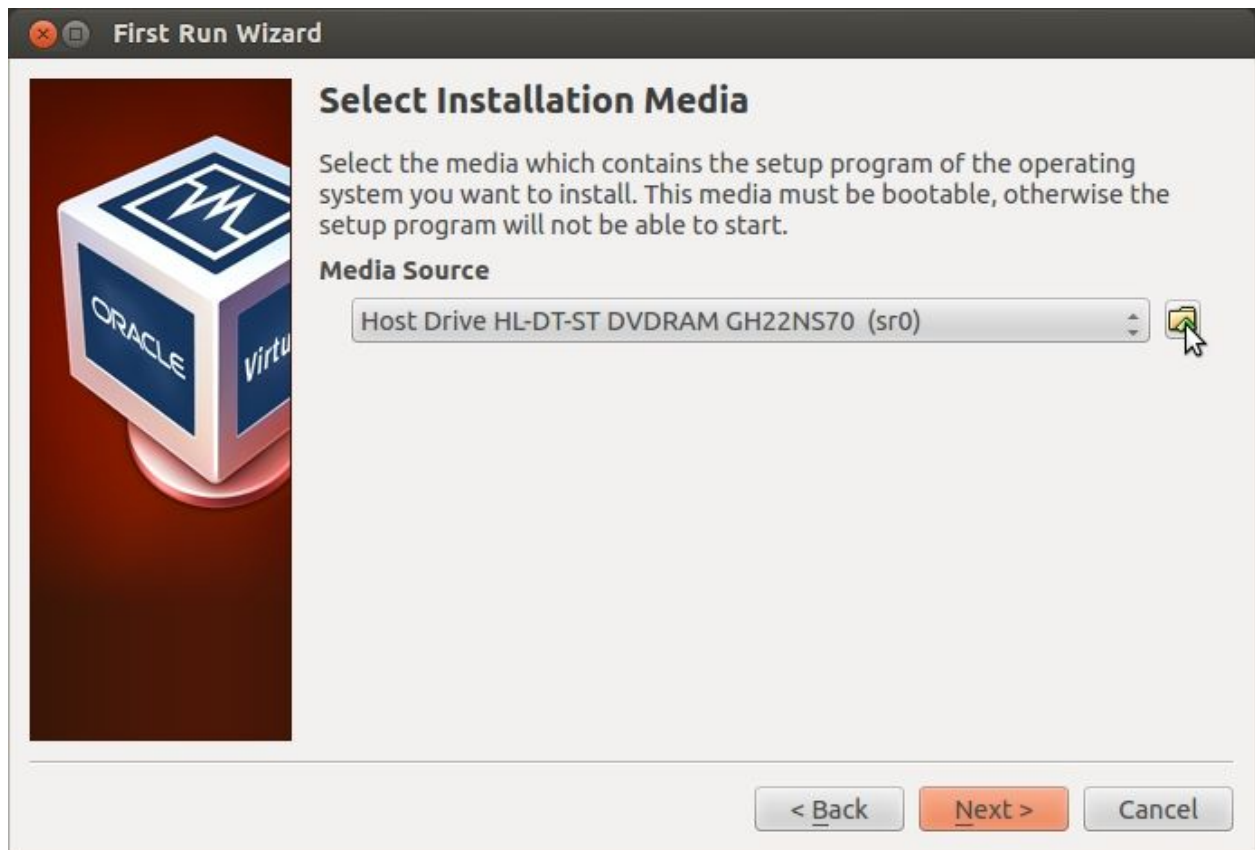
The "New Virtual Machine Wizard" will close and back to VirtualBox Manager. Select your Virtual Machine and click **Start** button.



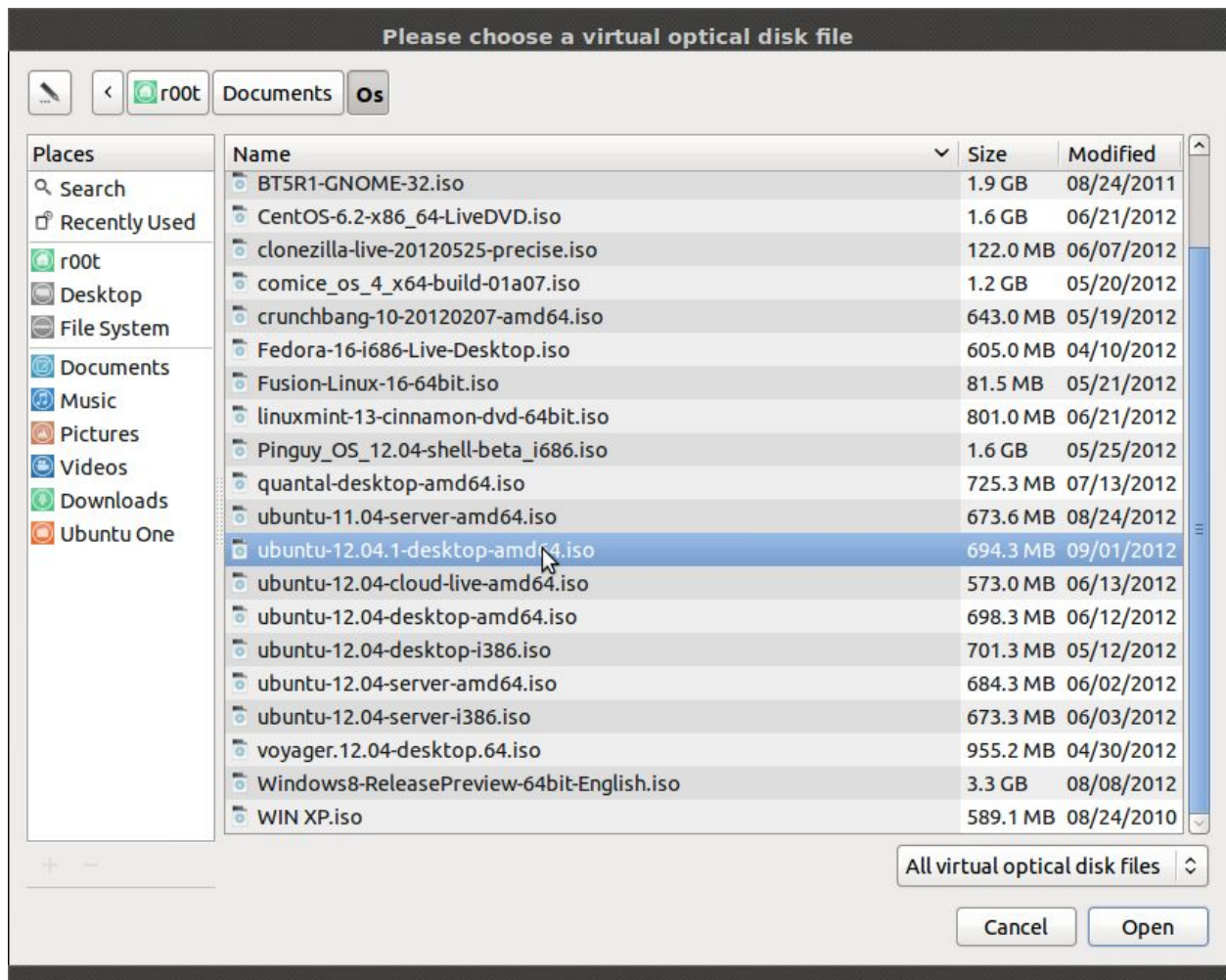
"First Run Wizard" will appear and click **Next** button.



Click at 'folder' icon and choose your Ubuntu iso directory.

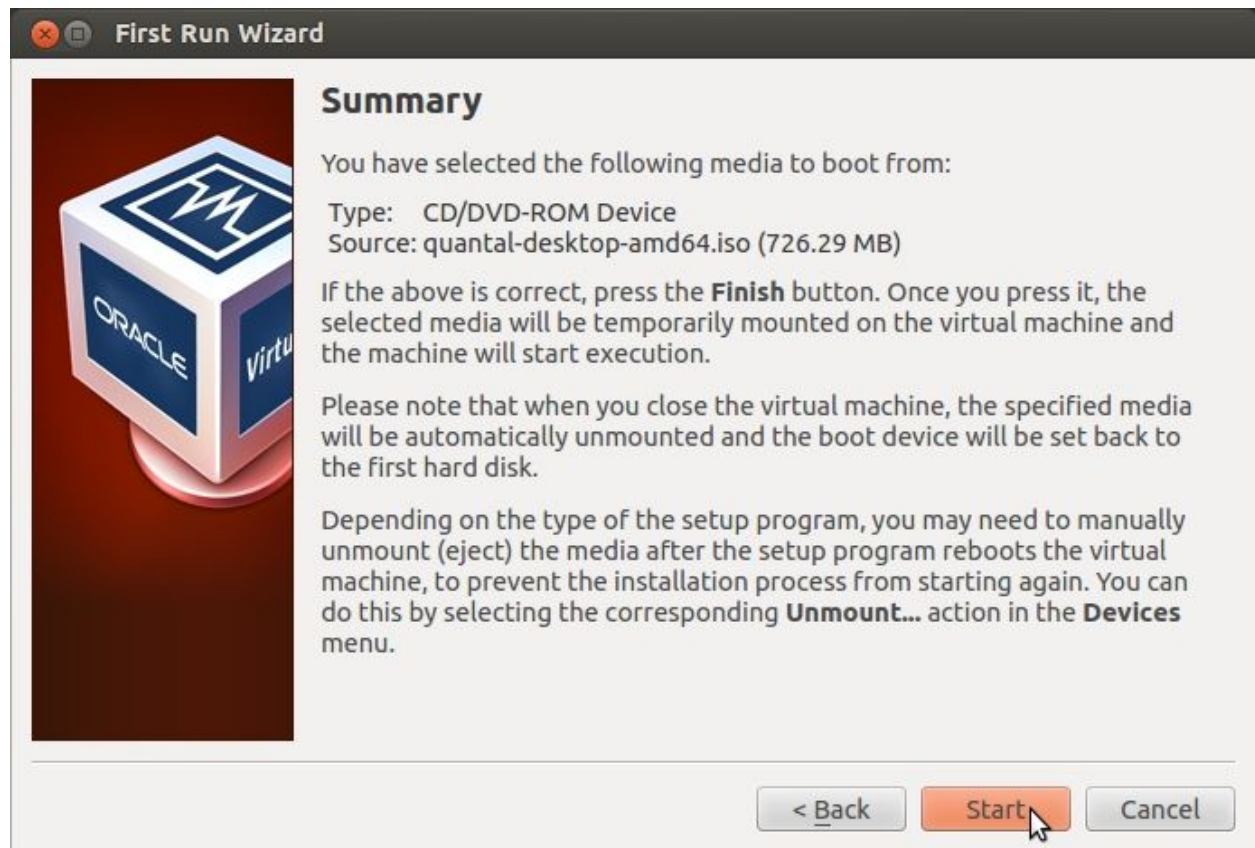


Select your Ubuntu iso file and click **Next** button.



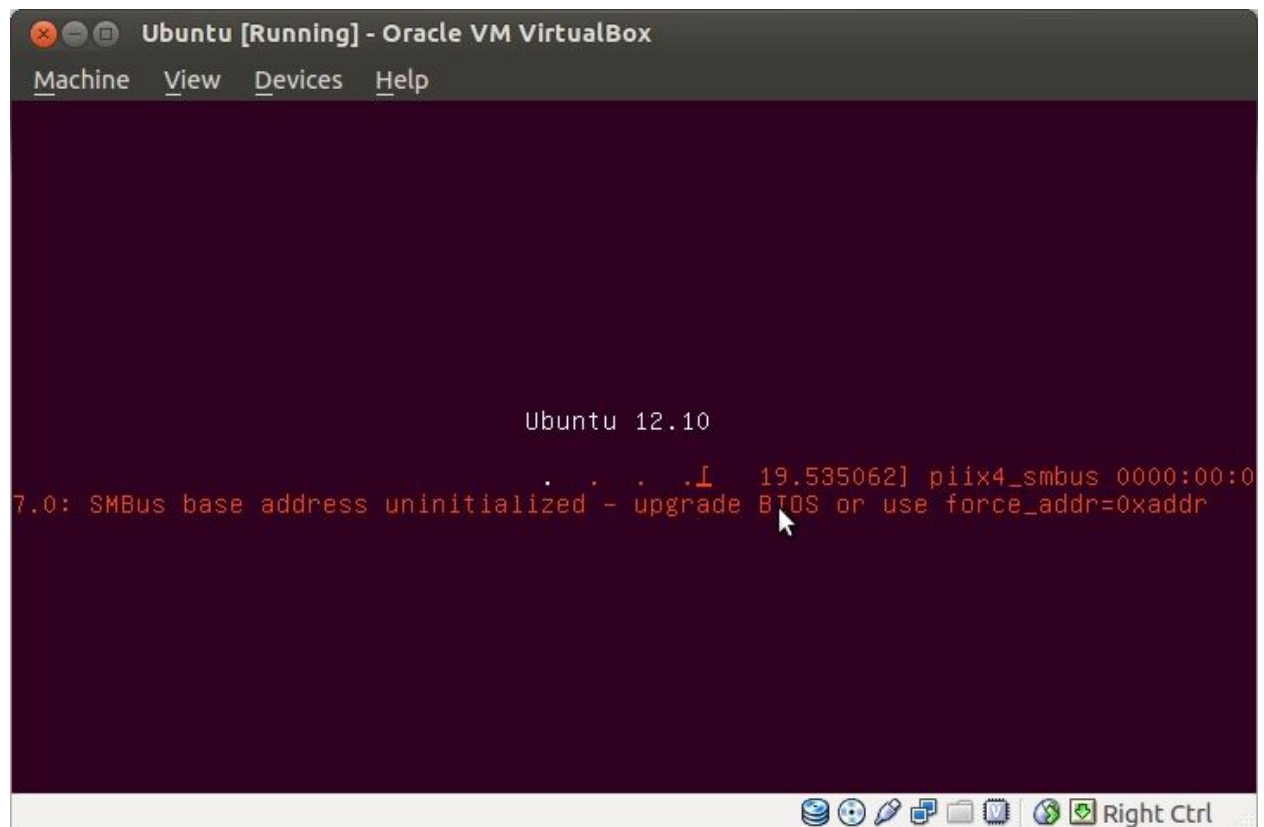
In 'Summary' box, click **Start** button.



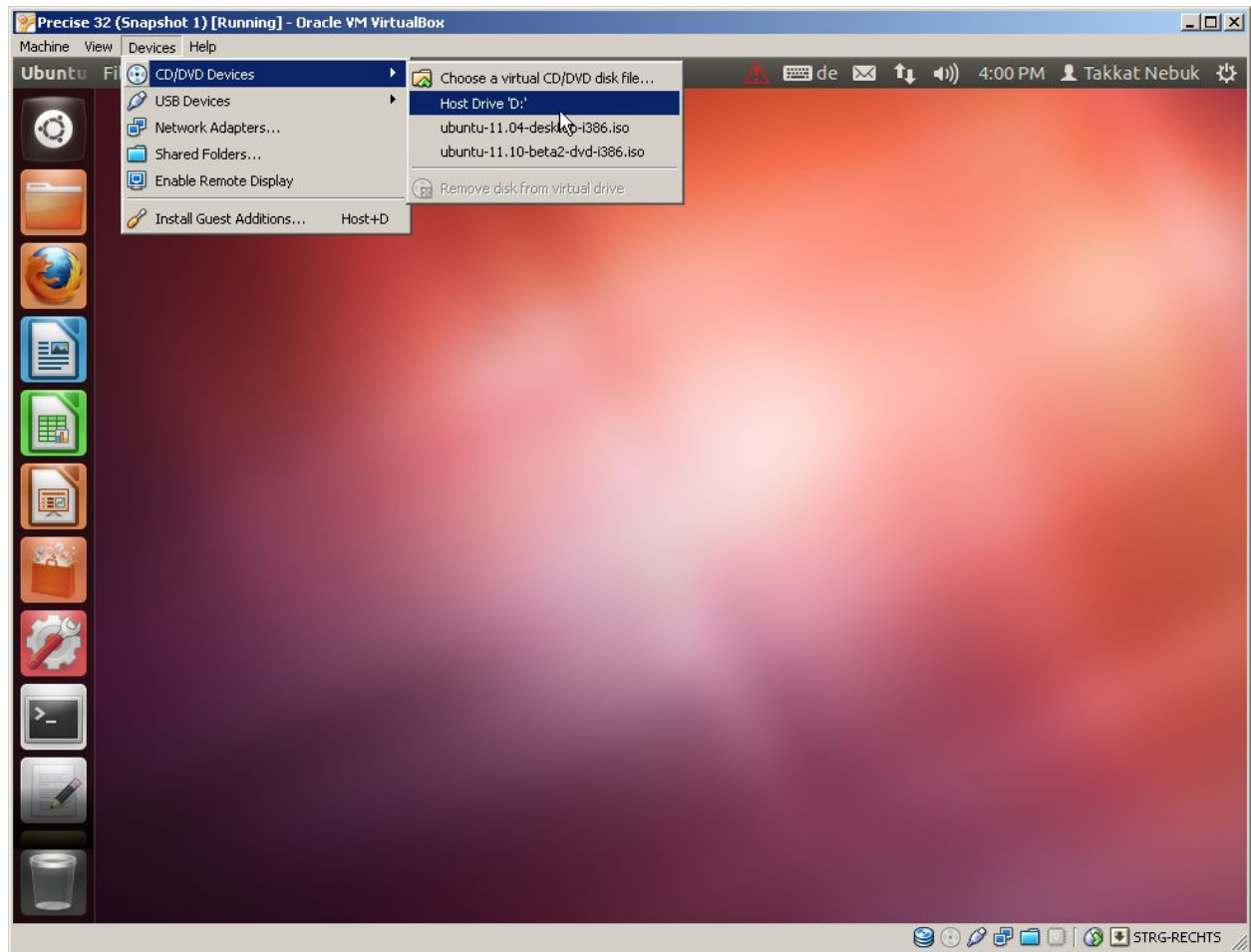


This screen will appear when it start boot.





After a successful installation we have to remove our installation .iso image from the virtual optical drive before we reboot. This can be done from the "Devices" menu or by removing the .iso from the VM settings:



## **17. Introduction to packet management system :**

**Given a set of RPM or DEB, how to build and maintain, serve packages over http or ftp. and also how do you configure client systems to access the package repository.**

**Package Management Basics: apt, yum, dnf, pkg**

### **Introduction**

Most modern Unix-like operating systems offer a centralized mechanism for finding and installing software. Software is usually distributed in the form of packages, kept in repositories. Working with packages is known as package management. Packages provide the basic components of an operating system, along with shared libraries, applications, services, and documentation.

A package management system does much more than one-time installation of software. It also provides tools for upgrading already-installed packages. Package repositories help to ensure that code has been vetted for use on your system, and that the installed versions of software have been approved by developers and package maintainers.

When configuring servers or development environments, it's often necessary look beyond official repositories. Packages in the stable release of a distribution may be out of date, especially where new or rapidly-changing software is concerned. Nevertheless,

package management is a vital skill for system administrators and developers, and the wealth of packaged software for major distributions is a tremendous resource.

This guide is intended as a quick reference for the fundamentals of finding, installing, and upgrading packages on a variety of distributions, and should help you translate that knowledge between systems.

## Package Management Systems: A Brief Overview

Most package systems are built around collections of package files. A package file is usually an archive which contains compiled binaries and other resources making up the software, along with installation scripts. Packages also contain valuable metadata, including their dependencies, a list of other packages required to install and run them.

While their functionality and benefits are broadly similar, packaging formats and tools vary by platform:

Operating System	Format	Tool(s)
Debian	.deb	apt, apt-cache, apt-get, dpkg
Ubuntu	.deb	apt, apt-cache, apt-get, dpkg
CentOS	.rpm	yum
Fedora	.rpm	dnf

FreeBSD	Ports, .txz	make, pkg
---------	-------------	-----------

In Debian and systems based on it, like Ubuntu, Linux Mint, and Raspbian, the package format is the .debfile. APT, the Advanced Packaging Tool, provides commands used for most common operations: Searching repositories, installing collections of packages and their dependencies, and managing upgrades. APT commands operate as a front-end to the lower-level dpkg utility, which handles the installation of individual .deb files on the local system, and is sometimes invoked directly.

Recent releases of most Debian-derived distributions include the apt command, which offers a concise and unified interface to common operations that have traditionally been handled by the more-specific apt-get and apt-cache. Its use is optional, but may simplify some tasks.

CentOS, Fedora, and other members of the Red Hat family use RPM files. In CentOS, yum is used to interact with both individual package files and repositories.

In recent versions of Fedora, yum has been supplanted by dnf, a modernized fork which retains most of yum's interface.

FreeBSD's binary package system is administered with the pkg command. FreeBSD also offers the Ports Collection, a local directory structure and tools which allow the user to fetch, compile, and install packages directly from source using Makefiles. It's usually much more convenient to use pkg, but occasionally a pre-compiled package is unavailable, or you may need to change compile-time options.

## Update Package Lists

Most systems keep a local database of the packages available from remote repositories. It's best to update this database before installing or upgrading packages. As a partial exception to this pattern, yum and dnf will check for updates before performing some operations, but you can ask them at any time whether updates are available.

System	Command
Debian / Ubuntu	<code>sudo apt-get update</code>
	<code>sudo apt update</code>
CentOS	<code>yum check-update</code>
Fedora	<code>dnf check-update</code>
FreeBSD Packages	<code>sudo pkg update</code>
FreeBSD Ports	<code>sudo portsnap fetch update</code>

## Upgrade Installed Packages

Making sure that all of the installed software on a machine stays up to date would be an enormous undertaking without a package system. You would have to track upstream changes and security alerts for hundreds of different packages. While a package

manager doesn't solve every problem you'll encounter when upgrading software, it does enable you to maintain most system components with a few commands.

On FreeBSD, upgrading installed ports can introduce breaking changes or require manual configuration steps. It's best to read `/usr/ports/UPDATING` before upgrading with `portmaster`.

System	Command	Notes
Debian / Ubuntu	<code>sudo apt-get upgrade</code>	Only upgrades installed packages, where possible.
	<code>sudo apt-get dist-upgrade</code>	May add or remove packages to satisfy new dependencies.
	<code>sudo apt upgrade</code>	Like <code>apt-get upgrade</code> .
	<code>sudo apt full-upgrade</code>	Like <code>apt-get dist-upgrade</code> .
CentOS	<code>sudo yum update</code>	
Fedora	<code>sudo dnf upgrade</code>	

FreeBSD Packages	sudo pkg upgrade	
FreeBSD Ports	less /usr/ports/UPDATING	Uses less to view update notes for ports (use arrow keys to scroll, press q to quit).
	cd /usr/ports/ports-mgmt/portmaster && sudo make install && sudo portmaster -a	Installs portmaster and uses it to update installed ports.

## Find a Package

Most distributions offer a graphical or menu-driven front end to package collections. These can be a good way to browse by category and discover new software. Often, however, the quickest and most effective way to locate a package is to search with command-line tools.

System	Command	Notes
Debian / Ubuntu	apt-cache search <b>search_string</b>	
	apt search <b>search_string</b>	
CentOS	yum search <b>search_string</b>	



Fedora	yum search all <code>search_string</code>	Searches all fields, including description.
	dnf search <code>search_string</code>	
FreeBSD Packages	dnf search all <code>search_string</code>	Searches all fields, including description.
	pkg search <code>search_string</code>	Searches by name.
	pkg search -f <code>search_string</code>	Searches by name, returning full descriptions.
FreeBSD Ports	pkg search -D <code>search_string</code>	Searches description.
	cd /usr/ports && make search name= <code>package</code>	Searches by name.
	cd /usr/ports && make search key= <code>search_string</code>	Searches comments, descriptions, and dependencies.

## View Info About a Specific Package

When deciding what to install, it's often helpful to read detailed descriptions of packages. Along with human-readable text, these often include metadata like version numbers and a list of the package's dependencies.

System	Command	Notes
Debian / Ubuntu	apt-cache show <b>package</b>	Shows locally-cached info about a package.
	apt show <b>package</b>	
	dpkg -s <b>package</b>	Shows the current installed status of a package.
CentOS	yum info <b>package</b>	Lists dependencies for a package.
	yum deplist <b>package</b>	
Fedora	dnf info <b>package</b>	Lists dependencies for a package.
	dnf repoquery --requires <b>package</b>	
FreeBSD Packages	pkg info <b>package</b>	Shows info for an installed package.

FreeBSD  
Ports

```
cd /usr/ports/category/port && cat  
pkg-descr
```

## Install a Package from Repositories

Once you know the name of a package, you can usually install it and its dependencies with a single command. In general, you can supply multiple packages to install simply by listing them all.

System	Command	Notes
Debian / Ubuntu	<code>sudo apt-get install <b>package</b></code>	
	<code>sudo apt-get install <b>package1</b> <b>package2</b> ...</code>	Installs all listed packages.
	<code>sudo apt-get install -y <b>package</b></code>	Assumes "yes" where apt would usually prompt to continue.
	<code>sudo apt install <b>package</b></code>	Displays a colored progress bar.
CentOS	<code>sudo yum install <b>package</b></code>	
	<code>sudo yum install <b>package1</b> <b>package2</b> ...</code>	Installs all listed packages.

Fedora	<code>sudo yum install -y package</code>	Assumes "yes" where yum would usually prompt to continue.
	<code>sudo dnf install package</code>	
	<code>sudo dnf install package1 package2 ...</code>	Installs all listed packages.
FreeBSD Package s	<code>sudo dnf install -y package</code>	Assumes "yes" where dnf would usually prompt to continue.
	<code>sudo pkg install package</code>	
	<code>sudo pkg install package1 package2 ...</code>	Installs all listed packages.
FreeBSD Ports	<code>cd /usr/ports/category/port &amp;&amp; sudo make install</code>	Builds and installs a port from source.

## Install a Package from the Local Filesystem

Sometimes, even though software isn't officially packaged for a given operating system, a developer or vendor will offer package files for download. You can usually retrieve these with your web browser, or via `curl` on the command line. Once a package is on the target system, it can often be installed with a single command.

On Debian-derived systems, dpkg handles individual package files. If a package has unmet dependencies, gdebi can often be used to retrieve them from official repositories.

On CentOS and Fedora systems, yum and dnf are used to install individual files, and will also handle needed dependencies.

System	Command	Notes
Debian / Ubuntu	<code>sudo dpkg -i package.deb</code>  <code>sudo apt-get install -y gdebi &amp;&amp; sudo gdebi package.deb</code>	Installs and uses gdebi to install package.deb and retrieve any missing dependencies.
CentOS	<code>sudo yum install package.rpm</code>	
Fedora	<code>sudo dnf install package.rpm</code>	
FreeBSD Packages	<code>sudo pkg add package.txz</code>	

```
sudo pkg add -f package.txz
```

Installs package even if already installed.

## Remove One or More Installed Packages

Since a package manager knows what files are provided by a given package, it can usually remove them cleanly from a system if the software is no longer needed.

System	Command	Notes
Debian / Ubuntu	sudo apt-get remove <b>package</b>	Removes unneeded packages.
	sudo apt remove <b>package</b>	
	sudo apt-get autoremove	
CentOS	sudo yum remove <b>package</b>	
Fedora	sudo dnf erase <b>package</b>	
FreeBSD Packages	sudo pkg delete <b>package</b>	

FreeBSD Ports	<code>sudo pkg autoremove</code>	Removes unneeded packages.
	<code>sudo pkg delete <b>package</b></code>	
	<code>cd /usr/ports/<b>path_to_port</b> &amp;&amp; make deinstall</code>	De-installs an installed port.

## The apt Command

Administrators of Debian-family distributions are generally familiar with apt-get and apt-cache. Less widely known is the simplified apt interface, designed specifically for interactive use.

Traditional Command	apt Equivalent
<code>apt-get update</code>	<code>apt update</code>
<code>apt-get dist-upgrade</code>	<code>apt full-upgrade</code>
<code>apt-cache search <b>string</b></code>	<code>apt search <b>string</b></code>
<code>apt-get install <b>package</b></code>	<code>apt install <b>package</b></code>

apt-get remove **package**

apt remove **package**

apt-get purge **package**

apt purge **package**

While apt is often a quicker shorthand for a given operation, it's not intended as a complete replacement for the traditional tools, and its interface may change between versions to improve usability. If you are using package management commands inside a script or a shell pipeline, it's a good idea to stick with apt-get and apt-cache.

## Get Help

In addition to web-based documentation, keep in mind that Unix manual pages (usually referred to as man pages) are available for most commands from the shell. To read a page, use man:

- man **page**

In man, you can navigate with the arrow keys. Press / to search for text within the page, and q to quit.

System	Command	Notes
Debian / Ubuntu	man apt-get	Updating the local package database and working with packages.



	man apt-cache	Querying the local package database.
	man dpkg	Working with individual package files and querying installed packages.
	man apt	Working with a more concise, user-friendly interface to most basic operations.
CentOS	man yum	
Fedora	man dnf	
FreeBSD Packages	man pkg	Working with pre-compiled binary packages.
FreeBSD Ports	man ports	Working with the Ports Collection.

## 18. Installing various software packages. Either the package is yet to be installed or an older version is existing. The student can practice installing the latest version. Of course, this might need Internet access.

### 18.1. Install samba and share files to windows

#### File Server

One of the most common ways to network Ubuntu and Windows computers is to configure Samba as a File Server. This section covers setting up a Samba server to share files with Windows clients.

The server will be configured to share files with any client on the network without prompting for a password.

#### Installation

The first step is to install the samba package. From a terminal prompt enter:

```
sudo apt install samba
```

That's all there is to it; you are now ready to configure Samba to share files.

#### Configuration

The main Samba configuration file is located in `/etc/samba/smb.conf`. The default configuration file has a significant number of comments in order to document various configuration directives.

1. First, edit the following key/value pairs in the `[global]` section of `/etc/samba/smb.conf`:

```
workgroup = EXAMPLE
```

```
...
```

```
security = user
```

The security parameter is farther down in the `[global]` section, and is commented by default. Also, change `EXAMPLE` to better match your environment.

2. Create a new section at the bottom of the file, or uncomment one of the examples, for the directory to be shared:

```
[share]
comment = Ubuntu File Server Share
path = /srv/samba/share
browsable = yes
guest ok = yes
read only = no
create mask = 0755
```

- comment: a short description of the share. Adjust to fit your needs.
- path: the path to the directory to share.
- This example uses /srv/samba/sharename because, according to the Filesystem Hierarchy Standard (FHS), [/srv](#) is where site-specific data should be served. Technically Samba shares can be placed anywhere on the filesystem as long as the permissions are correct, but adhering to standards is recommended.
- browsable: enables Windows clients to browse the shared directory using Windows Explorer.
- guest ok: allows clients to connect to the share without supplying a password.
- read only: determines if the share is read only or if write privileges are granted. Write privileges are allowed only when the value is no, as is seen in this example. If the value is yes, then access to the share is read only.
- create mask: determines the permissions new files will have when created.

3. Now that Samba is configured, the directory needs to be created and the permissions changed. From a terminal enter:

```
sudo mkdir -p /srv/samba/share
sudo chown nobody:nogroup /srv/samba/share/
```

The -p switch tells mkdir to create the entire directory tree if it doesn't exist.

4. Finally, restart the samba services to enable the new configuration:

```
sudo systemctl restart smbd.service nmbd.service
```

Once again, the above configuration gives all access to any client on the local network. From a Windows client you should now be able to browse to the Ubuntu file server and see the shared directory. If your client doesn't show your share automatically, try to access your server by its IP address, e.g. \\192.168.1.1, in a Windows Explorer window. To check that everything is working try creating a directory from Windows.

To create additional shares simply create new [dir] sections in /etc/samba/smb.conf, and restart Samba. Just make sure that the directory you want to share actually exists and the permissions are correct.

The file share named "[share]" and the path /srv/samba/share are just examples. Adjust the share and path names to fit your environment. It is a good idea to name a share after a directory on the file system. Another example would be a share name of [qa] with a path of /srv/samba/qa.

## 18.2. Install Common Unix Printing System(CUPS)

### CUPS - Print Server

The primary mechanism for Ubuntu printing and print services is the Common UNIX Printing System (CUPS). This printing system is a freely available, portable printing layer which has become the new standard for printing in most Linux distributions. CUPS manages print jobs and queues and provides network printing using the standard Internet Printing Protocol (IPP), while offering support for a very large range of printers, from dot-matrix to laser and many in between. CUPS also supports PostScript Printer Description (PPD) and auto-detection of network printers, and features a simple web-based configuration and administration tool.

### Installation

To install CUPS on your Ubuntu computer, simply use sudo with the apt command and give the packages to install as the first parameter. A complete CUPS install has many package dependencies, but they may all be specified on the same command line. Enter the following at a terminal prompt to install CUPS:

```
sudo apt install cups
```

Upon authenticating with your user password, the packages should be downloaded and installed without error. Upon the conclusion of installation, the CUPS server will be started automatically.

For troubleshooting purposes, you can access CUPS server errors via the error log file at:

```
/var/log/cups/error_log.
```

If the error log does not show enough information to troubleshoot any problems you encounter, the verbosity of the CUPS log can be increased by changing the LogLevel directive in the configuration file (discussed below) to "debug" or even "debug2", which logs everything, from the default of "info". If you make this change, remember to change it back once you've solved your problem, to prevent the log file from becoming overly large.

# Configuration

The Common UNIX Printing System server's behavior is configured through the directives contained in the file `/etc/cups/cupsd.conf`. The CUPS configuration file follows the same syntax as the primary configuration file for the Apache HTTP server, so users familiar with editing Apache's configuration file should feel at ease when editing the CUPS configuration file. Some examples of settings you may wish to change initially will be presented here.

Prior to editing the configuration file, you should make a copy of the original file and protect it from writing, so you will have the original settings as a reference, and to reuse as necessary. Copy the `/etc/cups/cupsd.conf` file and protect it from writing with the following commands, issued at a terminal prompt:

```
sudo cp /etc/cups/cupsd.conf /etc/cups/cupsd.conf.original
sudo chmod a-w /etc/cups/cupsd.conf.original
```

- **ServerAdmin:** To configure the email address of the designated administrator of the CUPS server, simply edit the `/etc/cups/cupsd.conf` configuration file with your preferred text editor, and add or modify the `ServerAdmin` line accordingly. For example, if you are the Administrator for the CUPS server, and your e-mail address is `'bjoy@somebigco.com'`, then you would modify the `ServerAdmin` line to appear as such:

```
ServerAdmin bjoy@somebigco.com
```

- **Listen:** By default on Ubuntu, the CUPS server installation listens only on the loopback interface at IP address `127.0.0.1`. In order to instruct the CUPS server to listen on an actual network adapter's IP address, you must specify either a hostname, the IP address, or optionally, an IP address/port pairing via the addition of a `Listen` directive. For example, if your CUPS server resides on a local network at the IP address `192.168.10.250` and you'd like to make it accessible to the other systems on this subnetwork, you would edit the `/etc/cups/cupsd.conf` and add a `Listen` directive, as such:

```
Listen 127.0.0.1:631      # existing loopback Listen
Listen /var/run/cups/cups.sock # existing socket Listen
Listen 192.168.10.250:631  # Listen on the LAN interface, Port 631 (IPP)
```

- In the example above, you may comment out or remove the reference to the Loopback address (`127.0.0.1`) if you do not wish `cupsd` to listen on that interface, but would rather have it only listen on the Ethernet interfaces of the Local Area Network (LAN). To enable listening for all network interfaces for which a certain hostname is bound, including the Loopback, you could create a `Listen` entry for the hostname `socrates` as such:

```
Listen socrates:631 # Listen on all interfaces for the hostname 'socrates'
```

or by omitting the Listen directive and using Port instead, as in:

```
Port 631 # Listen on port 631 on all interfaces
```

For more examples of configuration directives in the CUPS server configuration file, view the associated system manual page by entering the following command at a terminal prompt:

```
man cupsd.conf
```

Whenever you make changes to the `/etc/cups/cupsd.conf` configuration file, you'll need to restart the CUPS server by typing the following command at a terminal prompt:

```
sudo systemctl restart cups.service
```

## Web Interface

CUPS can be configured and monitored using a web interface, which by default is available at <http://localhost:631/admin>. The web interface can be used to perform all printer management tasks.

In order to perform administrative tasks via the web interface, you must either have the root account enabled on your server, or authenticate as a user in the `lpadmin` group. For security reasons, CUPS won't authenticate a user that doesn't have a password.

To add a user to the `lpadmin` group, run at the terminal prompt:

```
sudo usermod -aG lpadmin username
```

# References

<https://github.com/suniltt/mygit/blob/master/lect1.pdf>

<https://cyberspace709.wordpress.com/>

<https://help.ubuntu.com/>