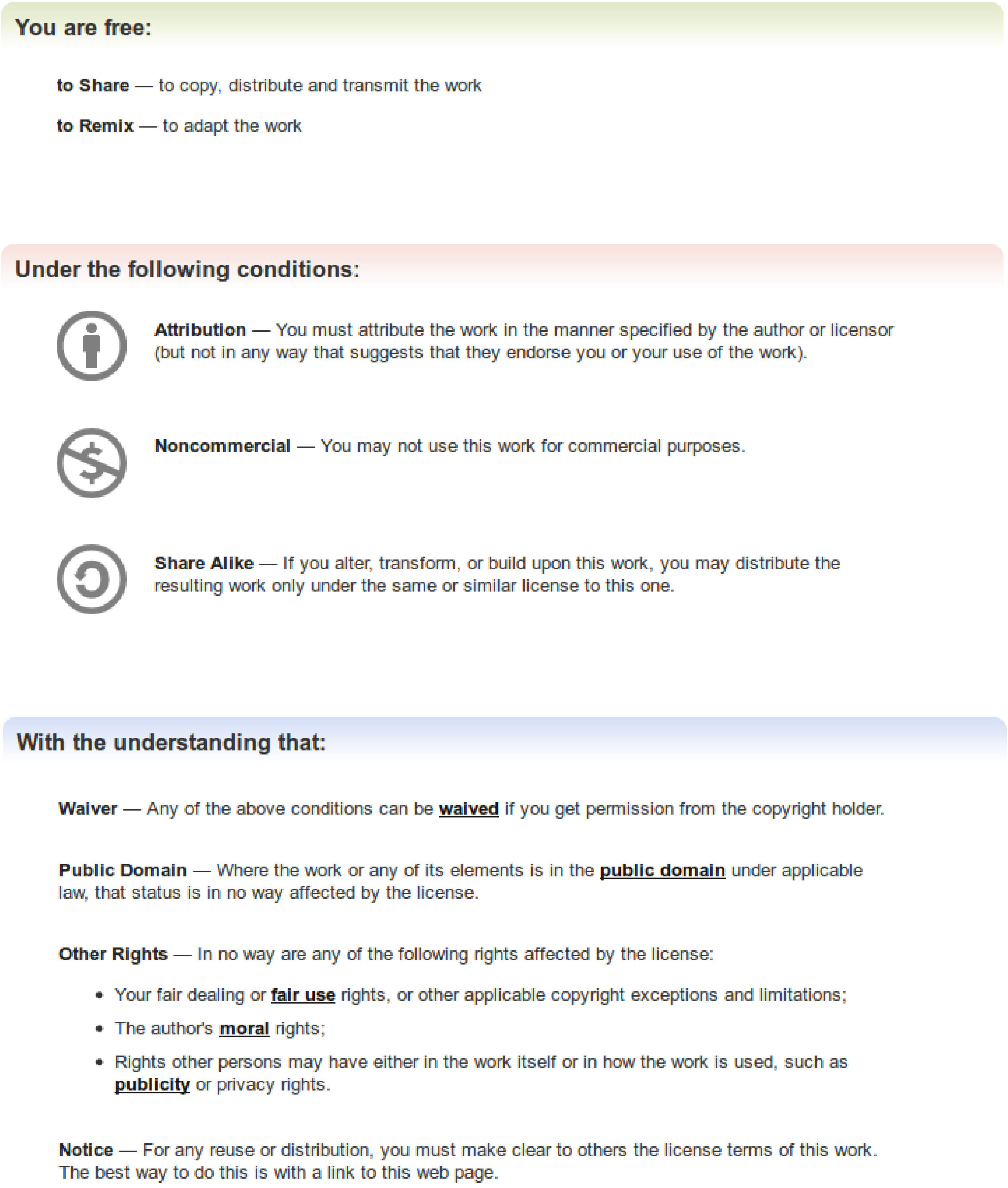
Beginner's Ubuntu Handbook

Released on 'September 2012' by Ankit Prateek
Proofread by Avinash Kumar
Revised Edition ‘May 2017’

CC Licence



You are free:

to Share — to copy, distribute and transmit the work

to Remix — to adapt the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Noncommercial — You may not use this work for commercial purposes.

Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

With the understanding that:

Waiver — Any of the above conditions can be **waived** if you get permission from the copyright holder.

Public Domain — Where the work or any of its elements is in the **public domain** under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

- Your fair dealing or **fair use** rights, or other applicable copyright exceptions and limitations;
- The author's **moral** rights;
- Rights other persons may have either in the work itself or in how the work is used, such as **publicity** or privacy rights.

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

Few Words From the Author

(Which most of you probably won't read)

I started writing this book for my cousins and friends who were patient enough to learn from me how to operate on linux. It started as a small basic linux guide and slowly turned into a study material.

This handbook is supposed to be distributed for free, anyone can suggest changes or make changes themselves and redistribute it for free. Commercialization is not allowed, only I can sell the printed copy of this book if I want to, sometime in the future. But even then I will make the pdf available for free. If you found this book helpful and wish to encourage my work, you can donate some money. I thought of asking the readers to rather donate the money to Johny Long's "Hackers For Charity" but my current condition of being involved with a Non-Governmental Organisation and another one that is a semi-Government Organisation for more than 7 months, I am so overworked and so underpaid, hardly paid in fact; if this goes on for a few more months soon I might have to approach some charitable trust to get two meals a day. So, I would really appreciate donations, or maybe you can just buy a coffee for me when you get a chance.

Anyways, there are a few sources that Iconstantly referred to while writing this book, wikipedia was helpful for checking on a lot of topics, some information which was new to me was copied exactly from it, then there was unixgeeks.
There might be a few more online resources I referred to which I don't remember exactly, I have been adding a lot of contents over a prolonged interval of time.

There are a few amazing resources I would recommend you to check out for troubleshooting and also to learn new stuffs from: howtogeek.com is a good resource, then use google a lot, there is ubuntuforums, ubuntugeek, noobslab etc. You will find a lot more on top google search results. You can check out my twitter account at ankit_appy or securitycrap, I follow a few of those linux resources, plus all the people I follow there are Awesome. I am constantly learning as much as I can from the resources there, plus I keep sharing what I am reading daily.

Feedbacks are greatly appreciated, I would love to add things or make changes if needed to make the book better. I left my twitter handle already, you can also leave feedbacks/suggestions on my email.

Thank You for choosing this book, and especially for reading this particular page above. :)
Hope You enjoy reading the book and may you find it useful.

Table of Contents

Linux

Linux is a free and open source software. It was in 1991 that Linus Benedict Torvalds begun the development of Linux. He is the God of all Geeks. The distribution and development of linux is done under the GNU Licence.

Linux is not as popular as Windows, not because it is difficult to use but because it does not comes as a default installation in Desktops and Laptops, another reason could be the lack of high-end gaming support. Linux has both GUI and CLI mode, much better than Windows. The CLI is very powerful in Linux. The GUI is also very easy to use, only difference is that it does not runs exe files, unlike windows.

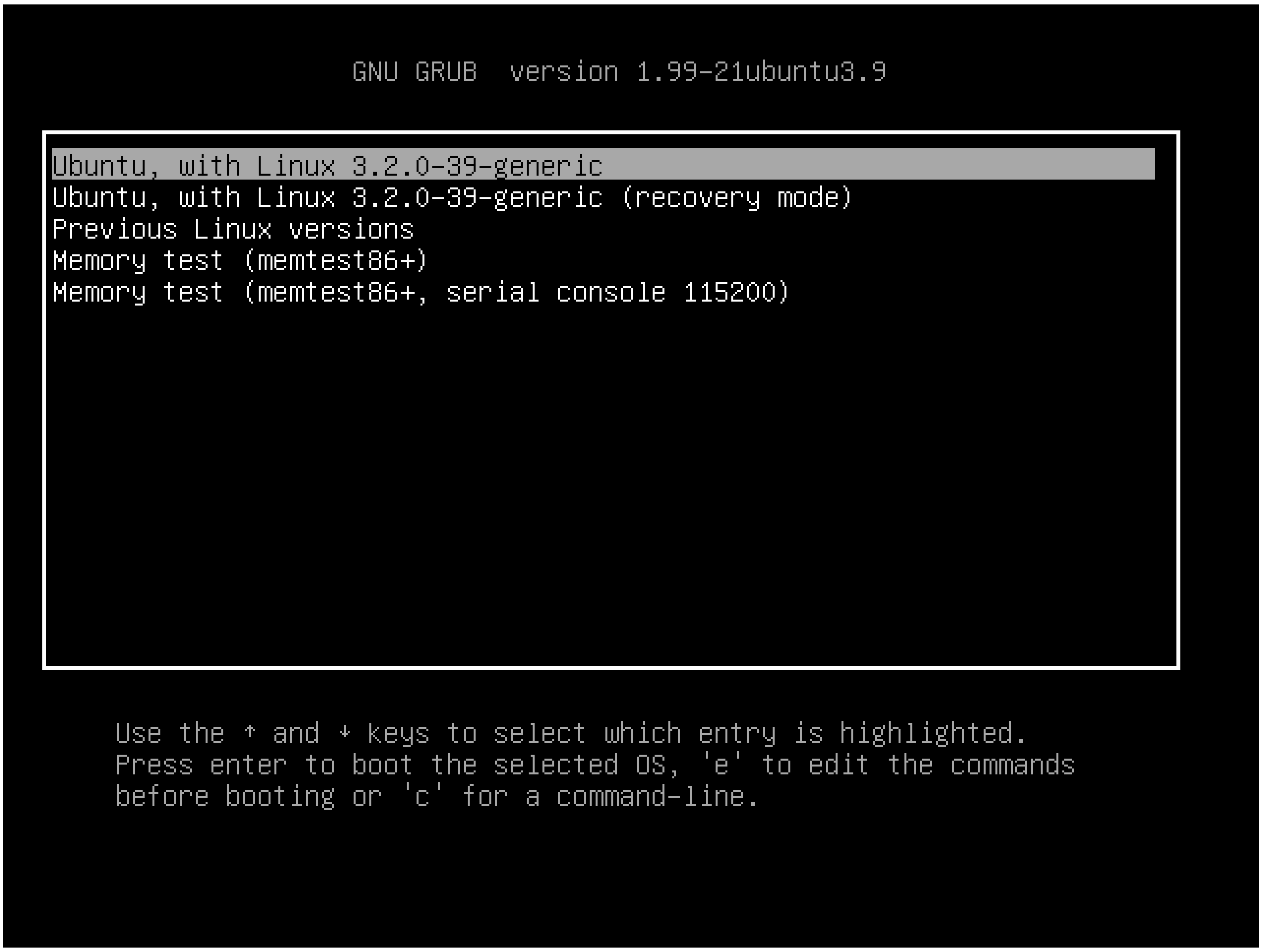
Before installation of Linux You must know that there is a bootloader that boots up the Operating System installed. In Windows the name of the bootloader is NTLDR(New Technology Loader) and in Linux its GRUB(Grand Unified Bootloader). Problem with NTLDR is that it does not recognizes any other OS other than Windows, seems like it was developed only with the vision of having single OS on PCs i.e. Windows.
But Grub recognizes Windows installation as well.

Boot Process

The entire **boot process** can be divided into 4 steps of:

First, the BIOS initialization, which then passes control to the IPL(Initial Program Loader). This is the second step of Boot loader. The IPL resides in the MBR(Master Boot record) in a very small space of maximum 446 bytes. The minimum specification for the successful boot loading of linux are: label, kernel location, OS root filesystem and the location of initial ramdisk(initrd), while the minimum specification for other OS are: boot device and label only.
Next, the third step is the kernel initialization followed by the final step where init starts and enters the run level.

On single boot where Linux is installed one might not be able to see the Grub menu, but it is there. To see the Grub menu, continue pressing Shift key while on boot up. There will be the linux-headers, recovery mode and memory test options. While on multiple boot the menu that appears on bootup is the Grub menu that asks you to choose between the OS to boot.
On a single boot, this is how the grub menu looks like:



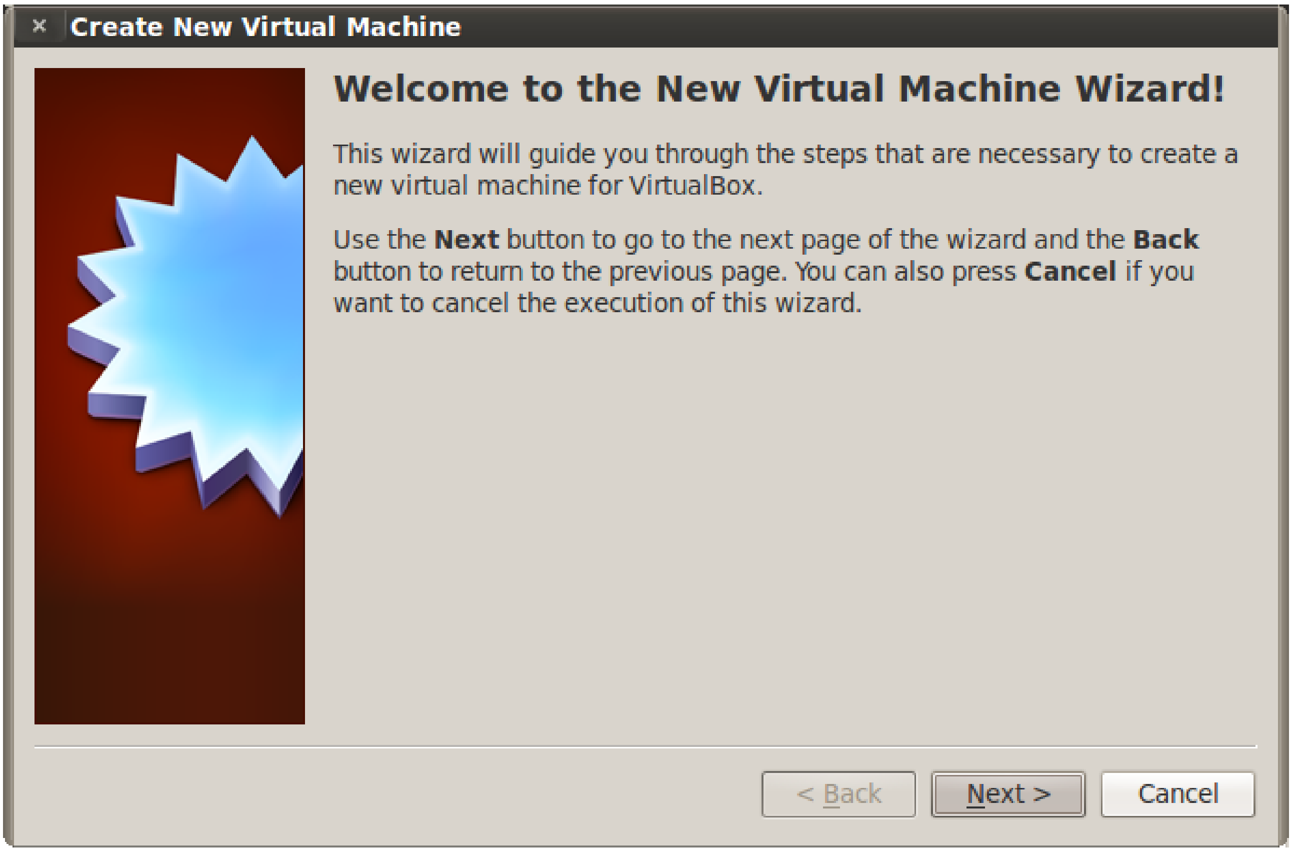
```
                    GNU GRUB  version 1.99-21ubuntu3.9

 Ubuntu, with Linux 3.2.0-39-generic
 Ubuntu, with Linux 3.2.0-39-generic (recovery mode)
 Previous Linux versions
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)

     Use the ↑ and ↓ keys to select which entry is highlighted.
     Press enter to boot the selected OS, 'e' to edit the commands
     before booting or 'c' for a command-line.
```

Configuring Virtualbox for Linux Installation

Here we will be installing Xubuntu 12.10 on virtualbox. The installation in main machine is done in a similar way. Xubuntu is Ubuntu with xfce desktop environment. Kubuntu is based on KDE, Lubuntu on LXDE. Simply put, it's all ubuntu with different Desktop themes.
First, get virtualbox from https://www.virtualbox.org/wiki/Downloads

Installing virtualbox is pretty simple. So skipping that, we will just move ahead to prepare the virtual environment for the linux installation.
Start virtualbox and click on "New" on top to setup a new machine. You will see the following window-

(On newer versions of virtualbox, this step has been removed.)



Create New Virtual Machine

Welcome to the New Virtual Machine Wizard!

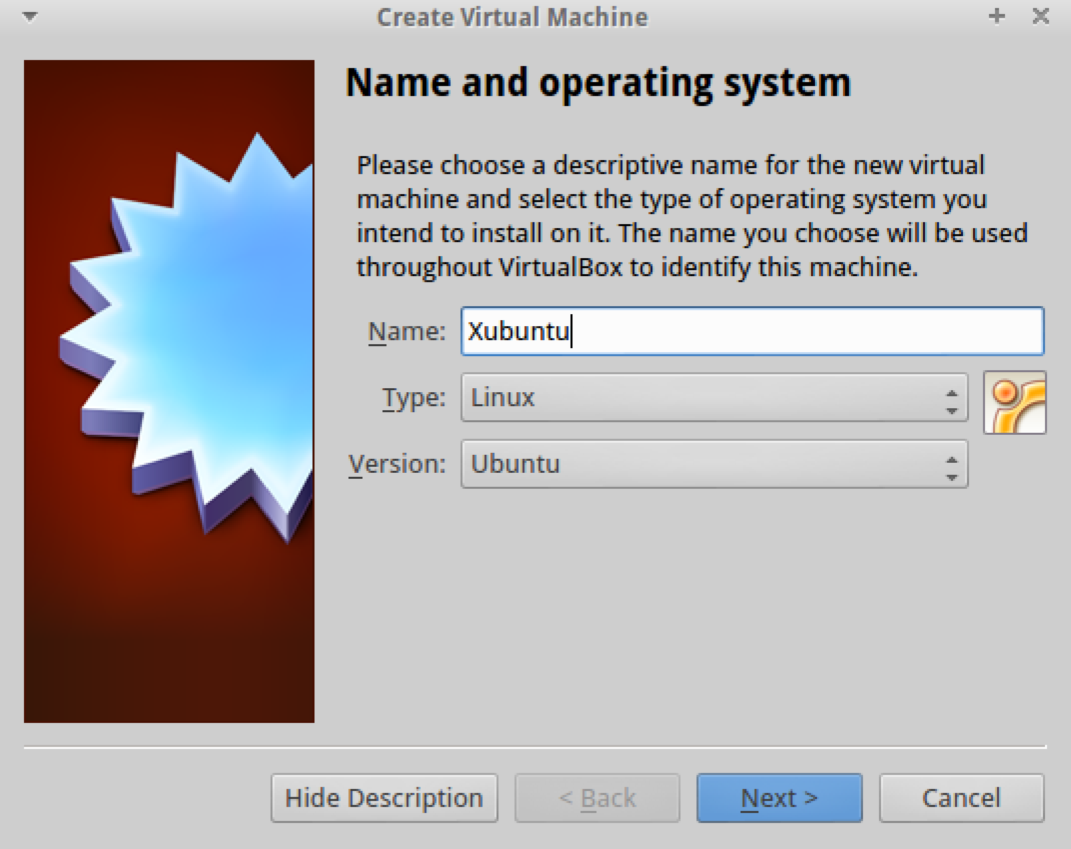
This wizard will guide you through the steps that are necessary to create a new virtual machine for VirtualBox.

Use the **Next** button to go to the next page of the wizard and the **Back** button to return to the previous page. You can also press **Cancel** if you want to cancel the execution of this wizard.

Create a new Virtual machine

1. Welcome Screen, do I need to mention you have to click on Next>?

Next window asks you the name of the OS and type, virtualbox has pre-configured settings for the OSes mentioned in the options, if you are installing an OS not mentioned in the options, don't worry, just choose the appropriate platform and name it, you will be prompted to change the settings.



Create Virtual Machine

Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: Xubuntu

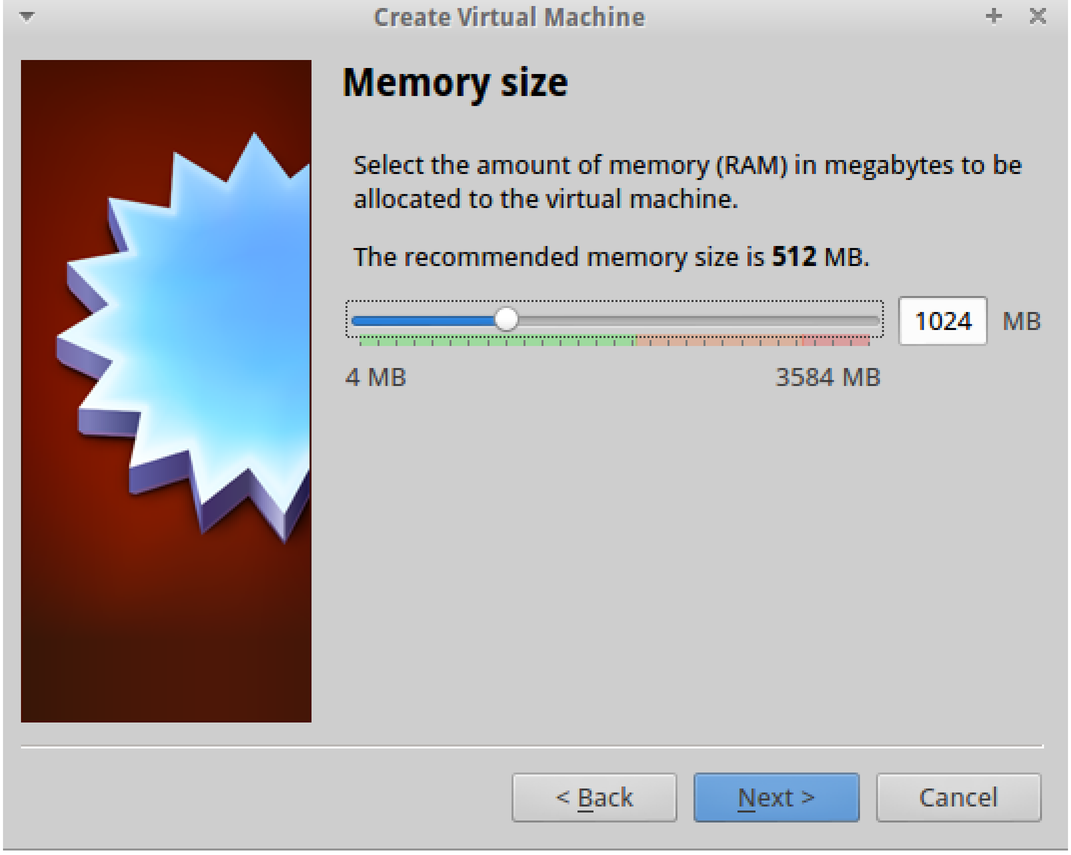
Type: Linux

Version: Ubuntu

Set Name & choose OS

On newer versions of virtual box this next snapshot is the first screen you will be presented with.

The next step will ask you to allot memory space to this virtual machine. Virtualbox recommends allotting less than half of the total RAM in host machine. For example, if you have a 2 GB RAM you can allot up to 1 GB memory for one virtual machine.



Create Virtual Machine

Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **512** MB.

1024 MB

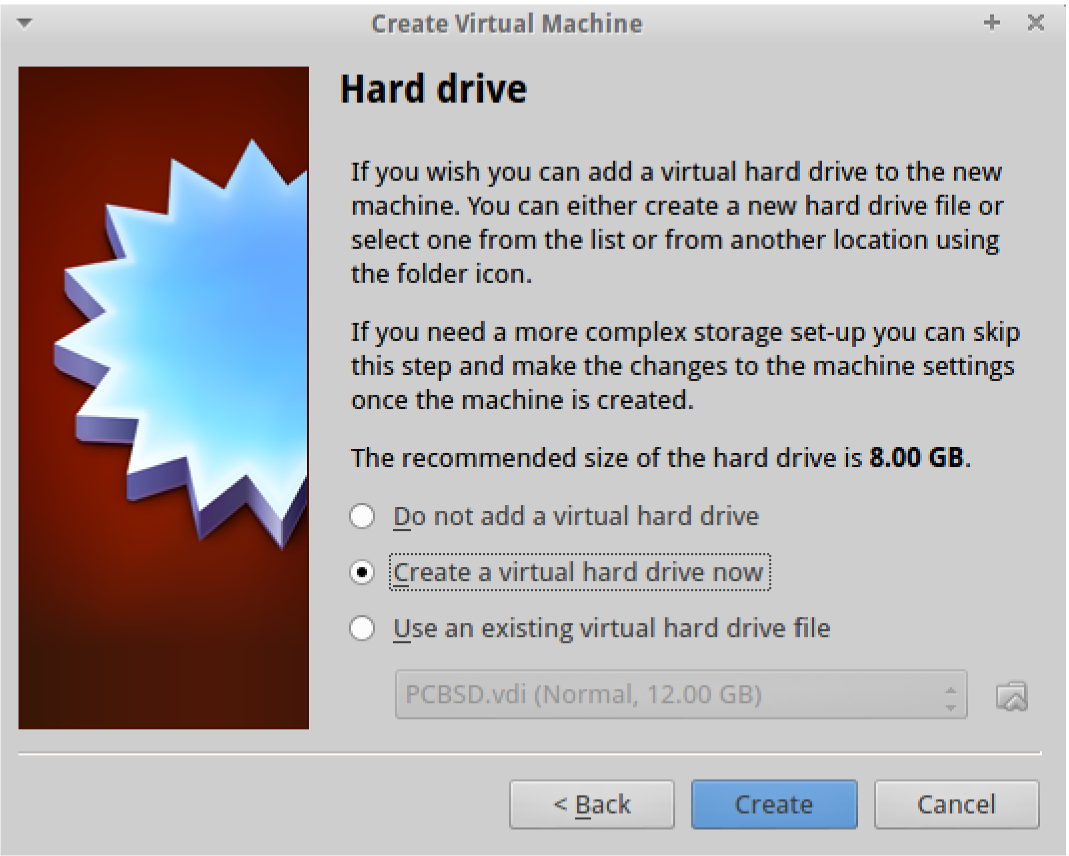
Allocating Memory

The default memory for Ubuntu is 512 MB, although version 10 can run fairly well with 256 MB RAM too.

I have a total of 4 GB RAM and so I am changing it to 1024 MB, depending on the RAM you have, allot memory accordingly.

Next step asks if you are creating a new start-up disk or you already have an existing disk(.vdi or .vmdk etc). A lot of virtual appliances are available online, from where you can download the virtual disk image and just point virtualbox to where its located and you can save the installation steps.

Here I am guessing it's your first time with virtualbox, so I will go with the "create a virtual hard drive now".



Hard drive

If you wish you can add a virtual hard drive to the new machine. You can either create a new hard drive file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard drive is **8.00 GB**.

○ Do not add a virtual hard drive

● Create a virtual hard drive now

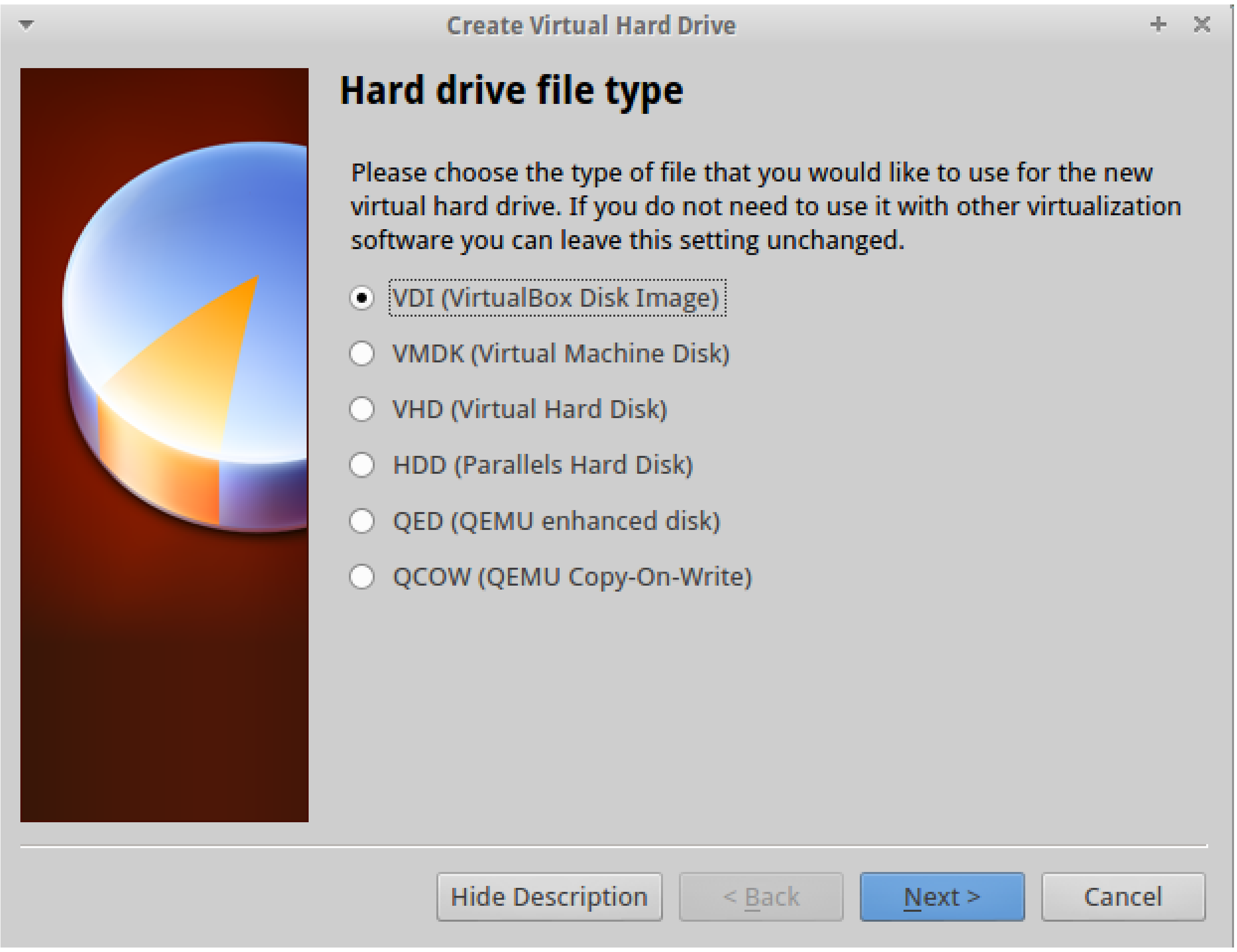
○ Use an existing virtual hard drive file

PCBSD.vdi (Normal, 12.00 GB)

Creating Virtual Hard Drive

This new hard disk does not requires a separate partition unlike the main installer, this solves the trouble of disk management and also removes the risk of deleting your main partition and losing all the existing data on the main machine. Point this .vdi file to be stored in any drive where you can spare some space, and virtualbox will treat this as a totally new partition for the OS being installed.

Lots of users accidentally end up destroying all their data while installing linux in the main hard drive, that is why we recommend installing linux on virtualbox first, once you get acquainted with the environment and workings, only then install it in the main drive.



Create Virtual Hard Drive

Hard drive file type

Please choose the type of file that you would like to use for the new virtual hard drive. If you do not need to use it with other virtualization software you can leave this setting unchanged.

● VDI (VirtualBox Disk Image)
○ VMDK (Virtual Machine Disk)
○ VHD (Virtual Hard Disk)
○ HDD (Parallels Hard Disk)
○ QED (QEMU enhanced disk)
○ QCOW (QEMU Copy-On-Write)

Next, you will be asked the filetype, default file type for virtual box is VDI.

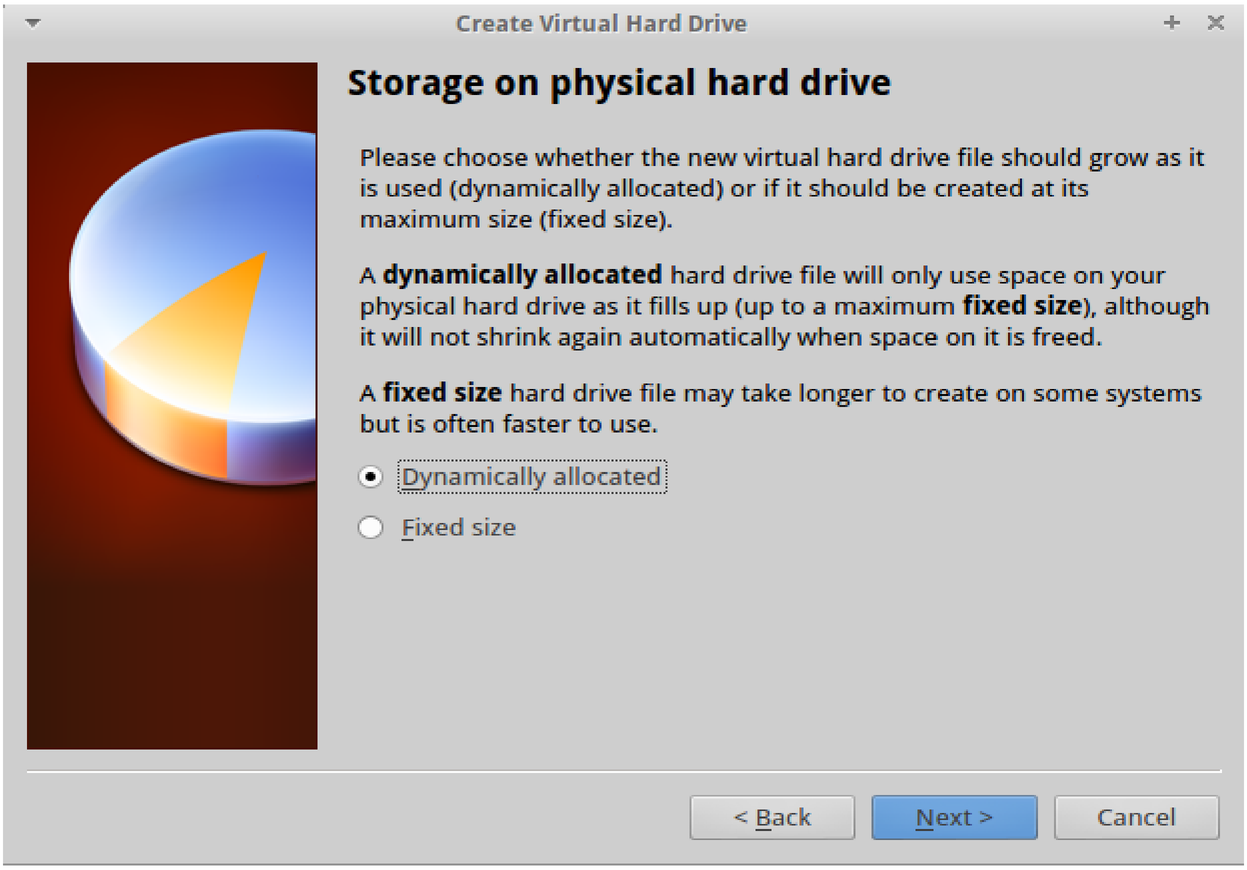
Hard Drive File Type

Please explore the other options later, find more about it on virtualbox help forums and experiment with the other options yourself.

Click next, and you will see a windows asking to select either a “dynamically expanding” or a “fixed storage”, with dynamically expanding type, the selected drive space will not be taken at once, like if you wish to give this OS a 10 GB of space then it can grow upto a maximum of 10 GB, initially it will take up a small size and will then expand according to the files you add to it, which can go upto 10 GB.

While with “fixed storage”, no matter how small the space it takes initially it will take up the entire 10GB space all at once from the host machine. Even if 10 GB space is not required at the moment it will use it anyways, so dynamically expandable option is better.

A little explanation on both the options is given there in the window too.



Create Virtual Hard Drive

Storage on physical hard drive

Please choose whether the new virtual hard drive file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

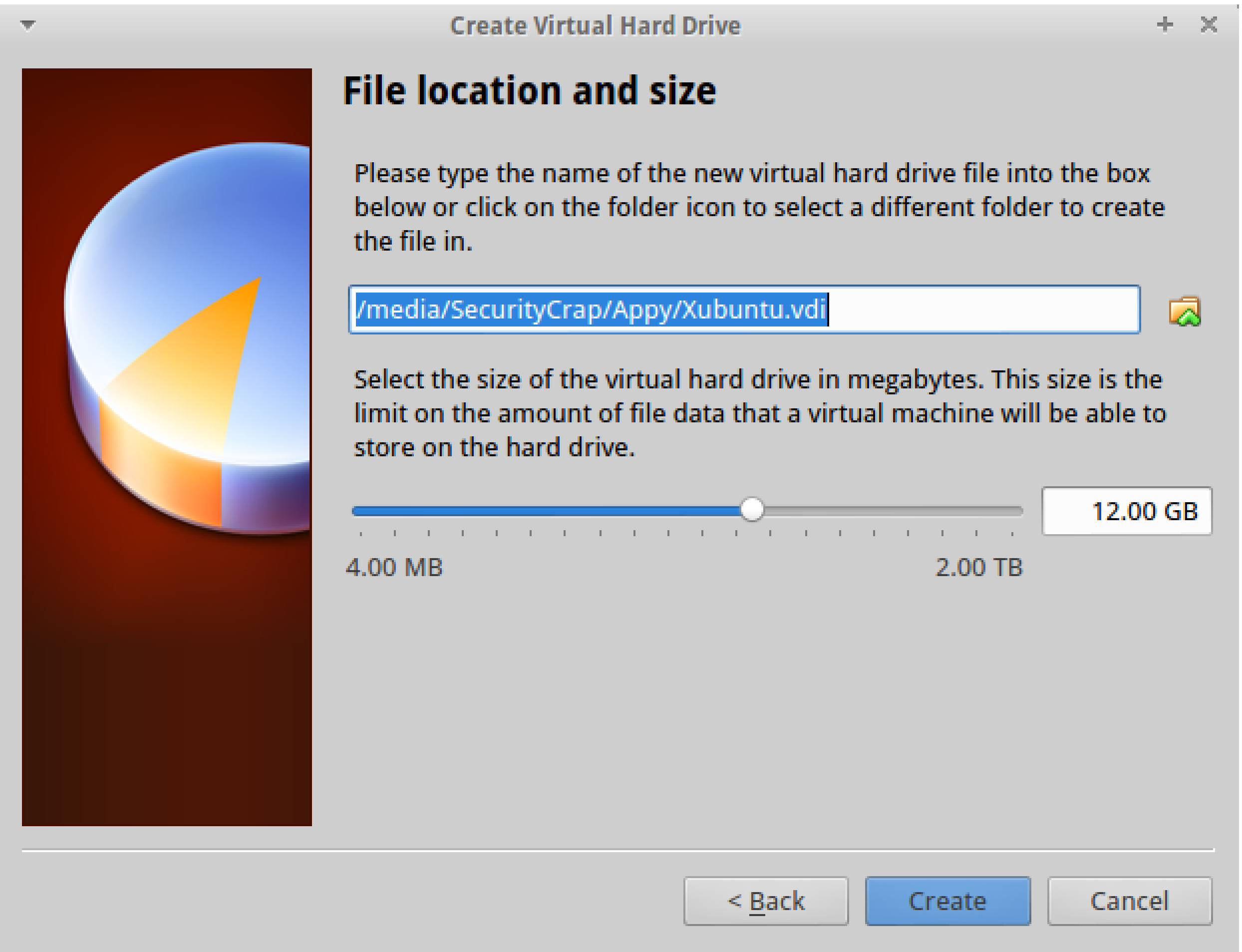
A **dynamically allocated** hard drive file will only use space on your physical hard drive as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard drive file may take longer to create on some systems but is often faster to use.

● Dynamically allocated

○ Fixed size

Type of Storage on hard Drive



Create Virtual Hard Drive

File location and size

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.

/media/SecurityCrap/Appy/Xubuntu.vdi

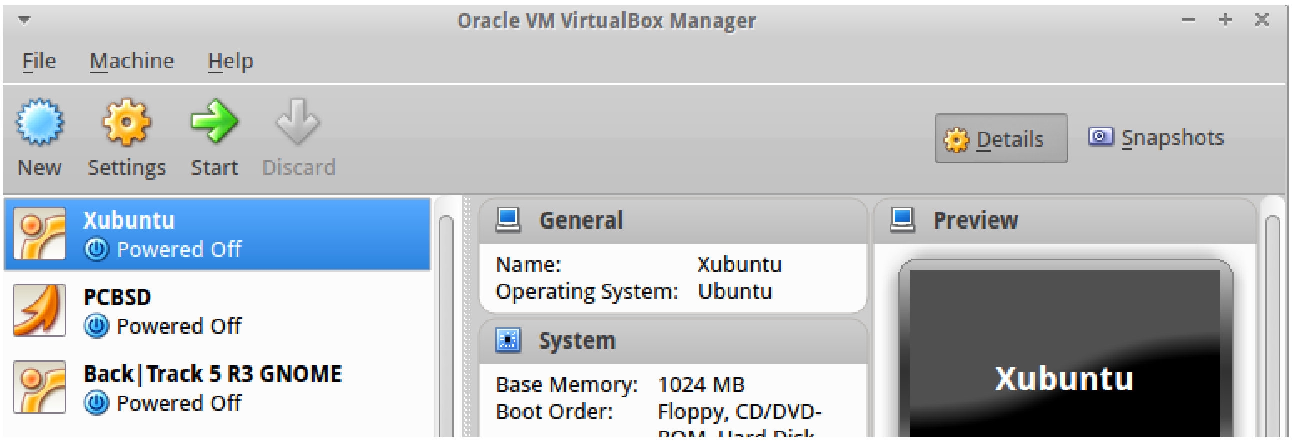
Select the size of the virtual hard drive in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard drive.

12.00 GB

File Location & Size of Hard Drive

Next you are asked to select the location where you want this new OS disk to be and how much hard disk space you can spare. It will show the default OS name and default space, change accordingly, for long term use give 10 GB at least for Ubuntu, although it can run in 4.5 GB too.
Next, you will either be presented with a summary of the settings or on newer versions of virtualbox the create button will skip the summary and just make the changes according to the settings chosen so far.

It's after now that all the changes will take place, if you need to change something it's the last chance to go back and change before changes are made. Click "Create" to finish configuring virtualbox. A newly created OS "Xubuntu" will now appear on the main window of Virtualbox as shown in the figure below:



Name: Xubuntu
Operating System: Ubuntu

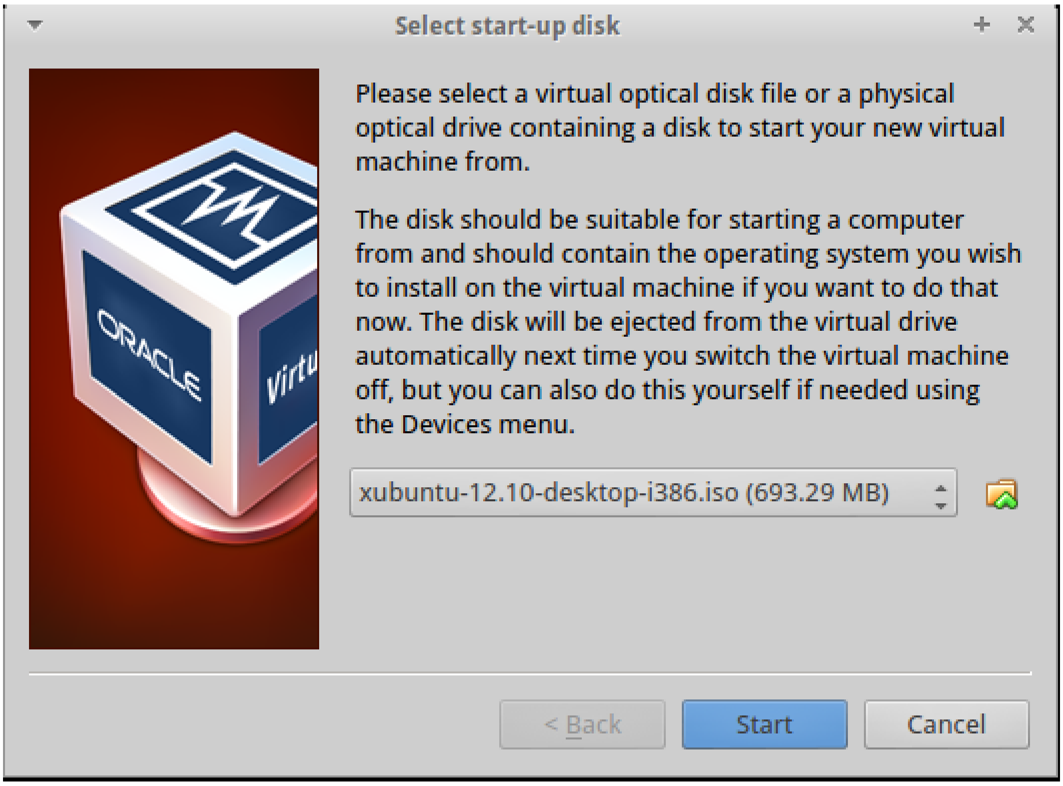
Base Memory: 1024 MB
Boot Order: Floppy, CD/DVD-ROM, Hard Disk

Select Xubuntu and click on "Start".

INSTALLATION

You will be presented with a window asking to specify the location of the installer disk, it can be an iso file on your drive or installation DVD in your optical drive.

I have pointed the wizard to the location of the xubuntu disk file downloaded from the official xubuntu website. Click on "Start" and the figure below is the first screen you are presented with:



Select start-up disk

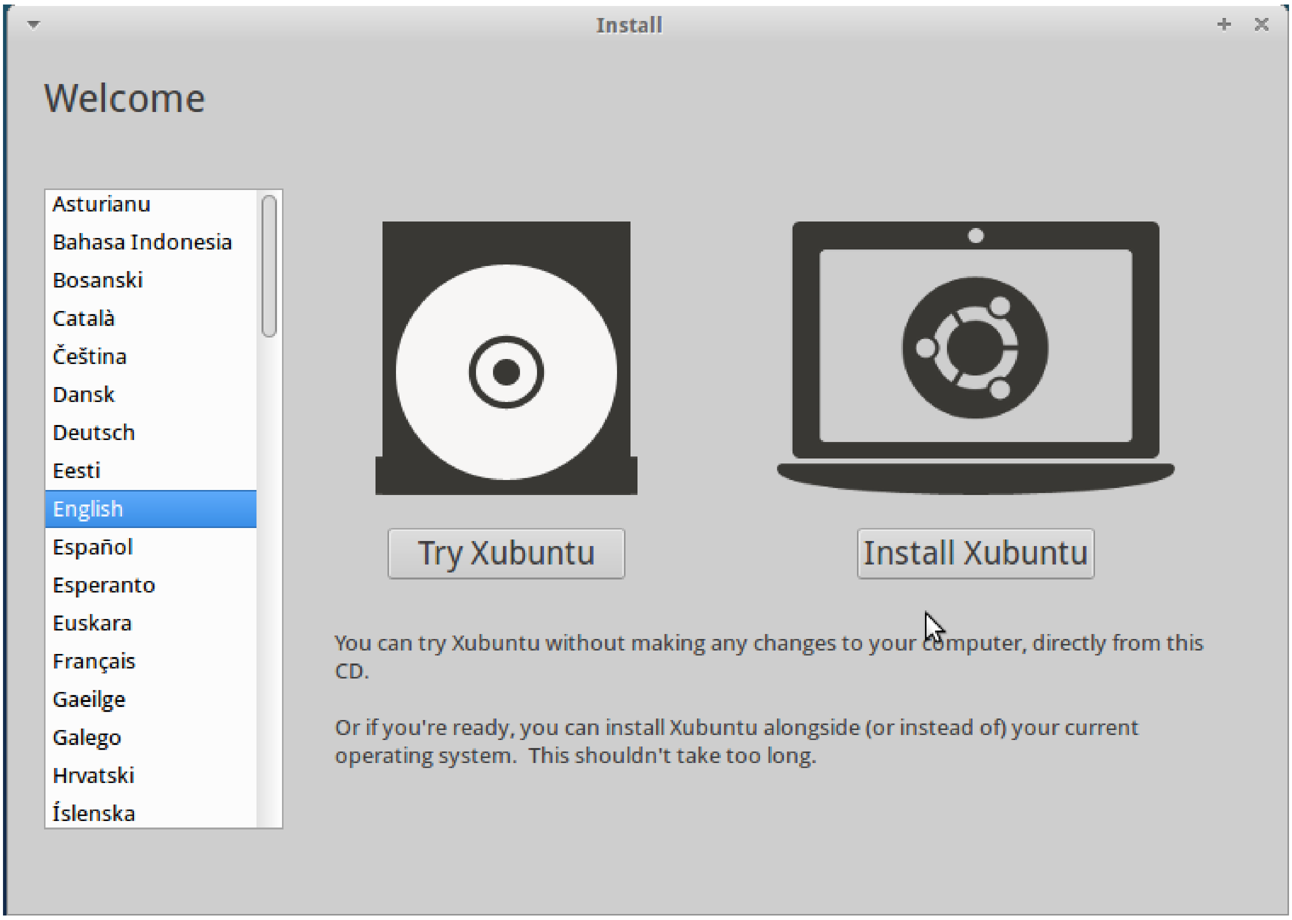
Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

xubuntu-12.10-desktop-i386.iso (693.29 MB)

Select start-up disk

And, here is your welcome screen:



Install

Welcome

You can try Xubuntu without making any changes to your computer, directly from this CD.

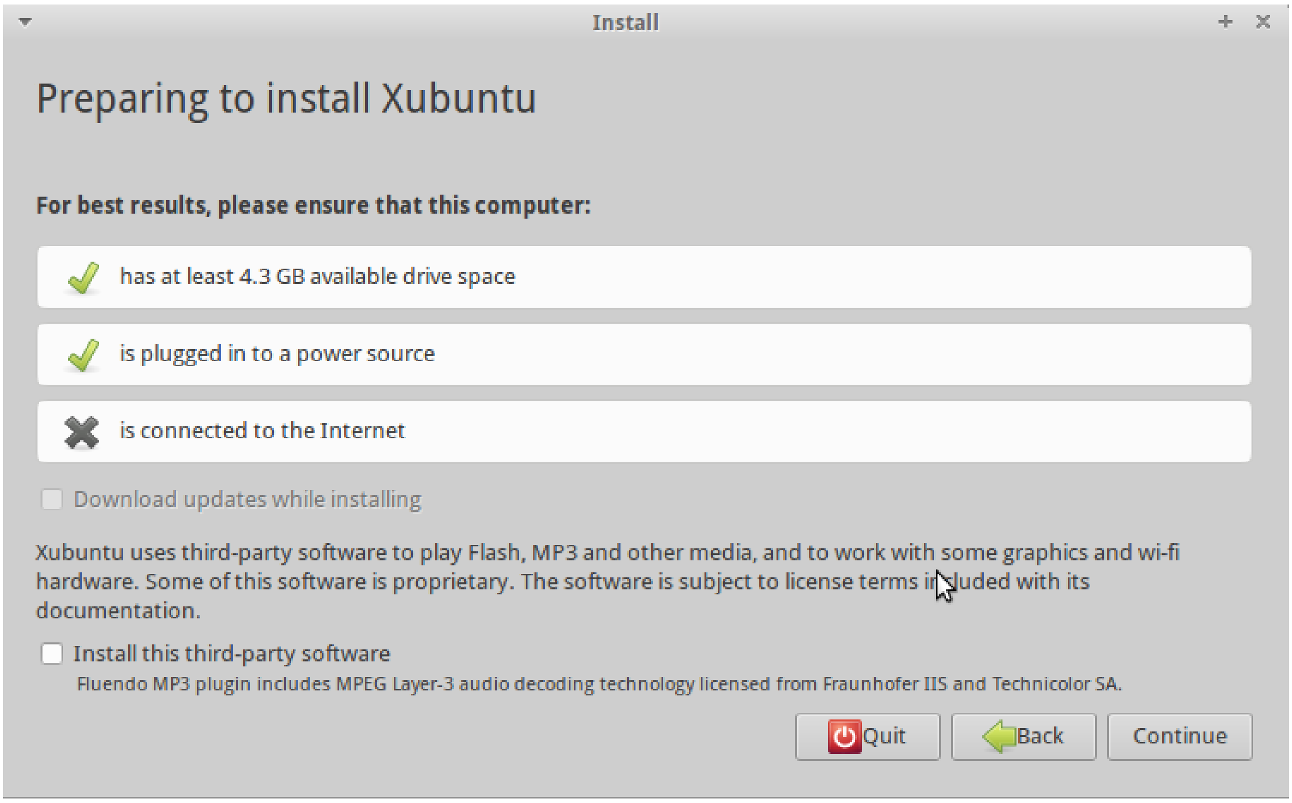
Or if you're ready, you can install Xubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

Welcome Screen

The “Try Xubuntu” option boots up the OS from installation media, without making any changes to the disk you can work on it, explore all the options and features. This is the beauty of Live Distros. Later in the book, we will learn how to create a bootable Linux USB, with that we can plug in the USB drive on any machine, boot it up from the USB and there we can have our own portable linux distro.

Now, since we have already configured virtualbox for installation, we will move to the next step by clicking on “Install Xubuntu”.

On the left side, the default language is set to “English”, change it to something you are more comfortable with.



Preparing to install Xubuntu

For best results, please ensure that this computer:

✓ has at least 4.3 GB available drive space

✓ is plugged in to a power source

✗ is connected to the Internet

□ Download updates while installing

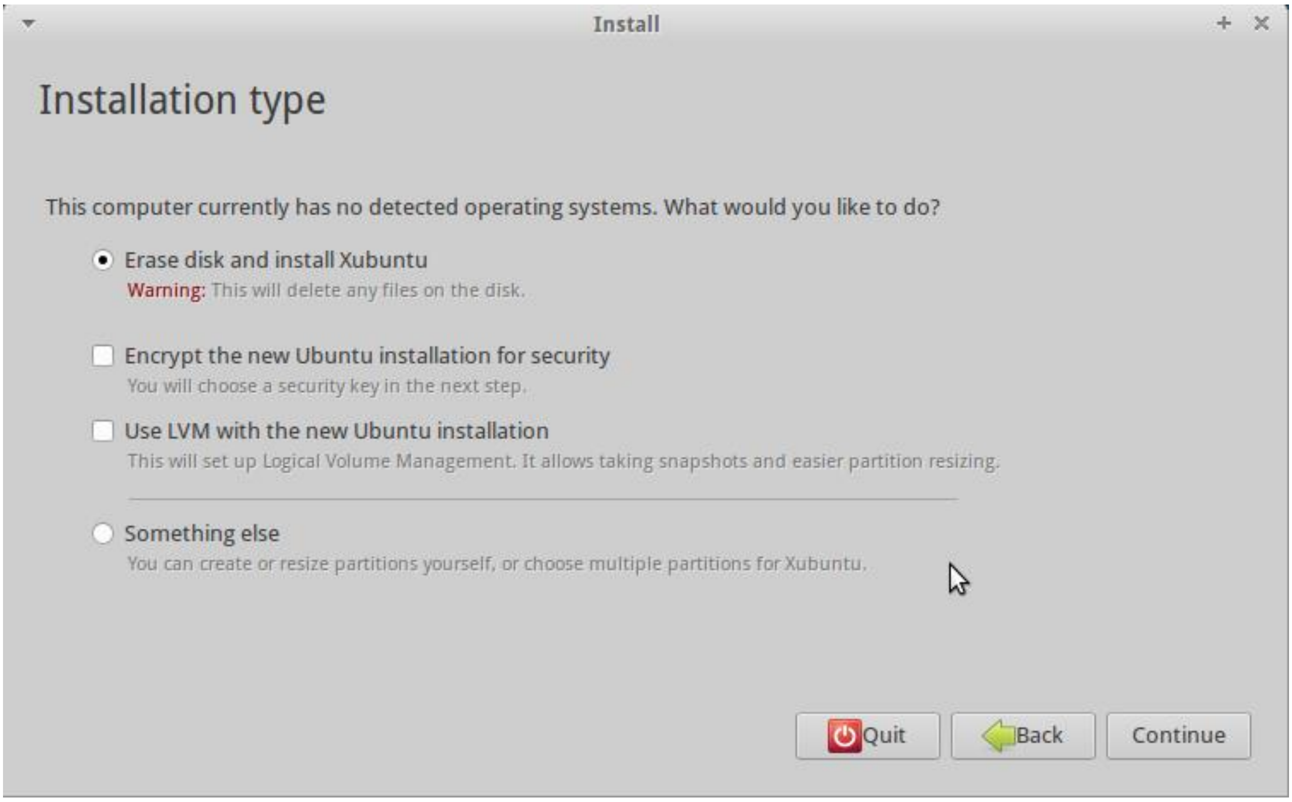
Xubuntu uses third-party software to play Flash, MP3 and other media, and to work with some graphics and wi-fi hardware. Some of this software is proprietary. The software is subject to license terms included with its documentation.

□ Install this third-party software

Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

Preparing Installation

The installer then verifies the existing conditions of the disk before proceeding with the installation.
The next step is the most important one if you are installing Ubuntu in Your main hard disk. This is the place where most new users end up losing all their data. But since this is being installed on virtualbox, there is no risk of losing any data on the disk drive.



Install

Installation type

This computer currently has no detected operating systems. What would you like to do?

● Erase disk and install Xubuntu
Warning: This will delete any files on the disk.

□ Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

□ Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

○ Something else
You can create or resize partitions yourself, or choose multiple partitions for Xubuntu.

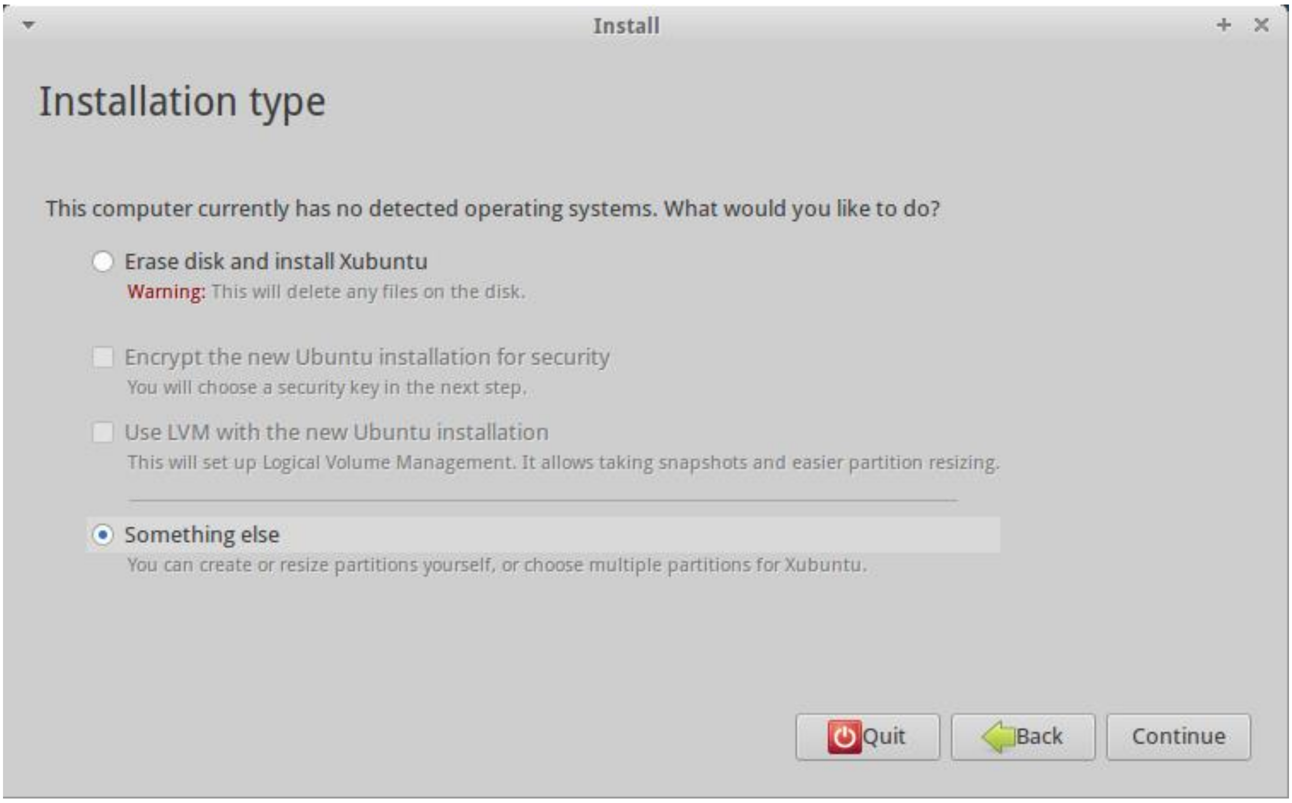
Installation Type

Here, the default option "Erase disk and install Ubuntu" will only erase the newly created .vdi file which is not even a partition, just a file acting like a partition for Ubuntu. That is the beauty of virtualization, and that is why we recommend using Ubuntu on virtualbox for starters.

The next option "Encrypt the new Ubuntu installation for security" was added in the 12.10 version which is a pretty good feature, after the easy basic installation we will check that out too.

Explaining the "Use LVM..." option is beyond the scope of this book, installation is not difficult but the concept is, a little. Although, if later you wish to learn more about advanced linux, you will find this option very useful for new space management. Simply put, the LVM feature helps a lot in adding more storage to an already installed linux machine, when needed in future.

So, all that being discussed; you can either choose the default option or the last one "Something else". I'd recommend you to make it a habit of always going with the "Something Else" option to reduce the risk of losing existing files on the drive.



Installation type

This computer currently has no detected operating systems. What would you like to do?

○ Erase disk and install Xubuntu
Warning: This will delete any files on the disk.

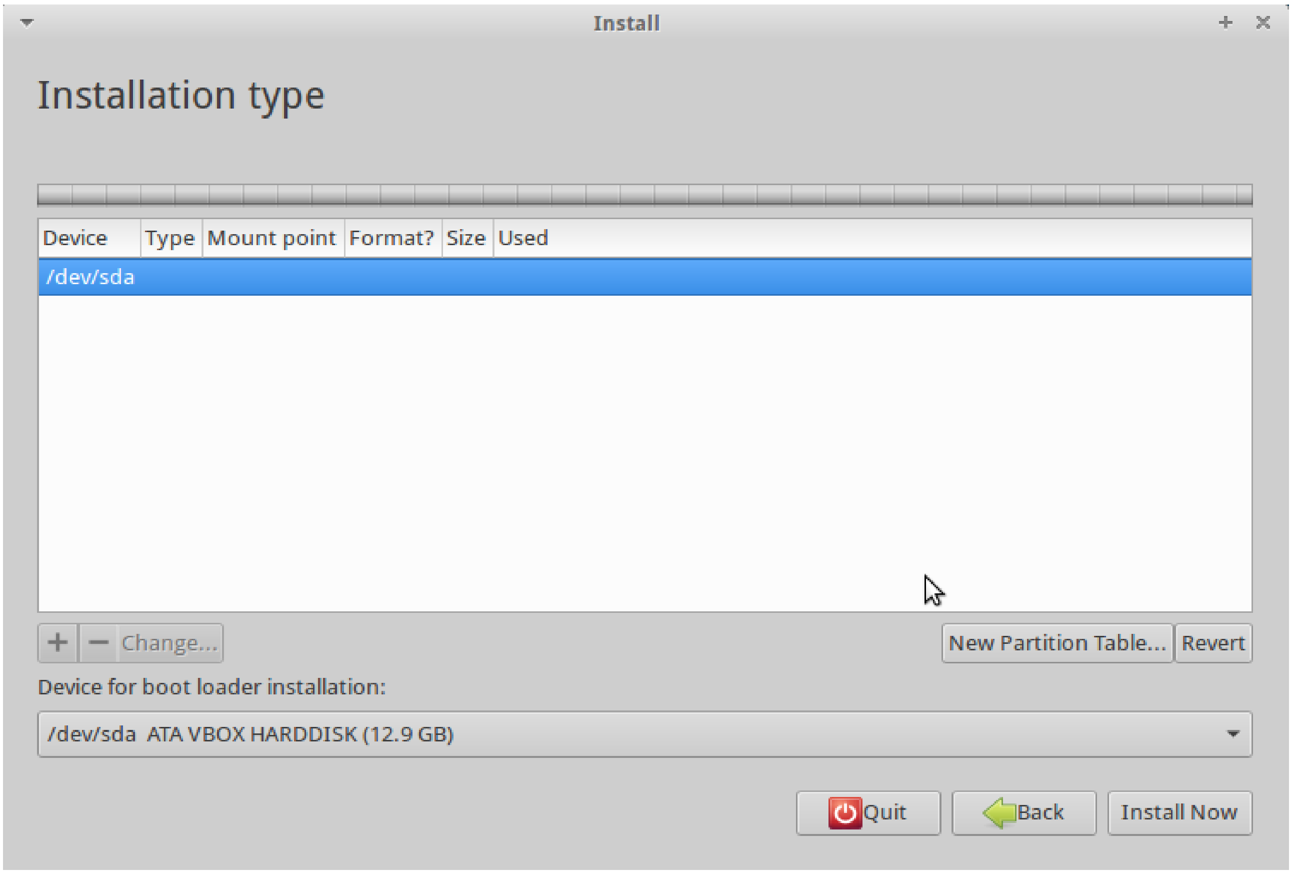
□ Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

□ Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

● Something else
You can create or resize partitions yourself, or choose multiple partitions for Xubuntu.

Always choose “something else” option

Next, you will see a window like the one below: a new hard disk, just like we configured virtualbox for Ubuntu installation.



Installation type

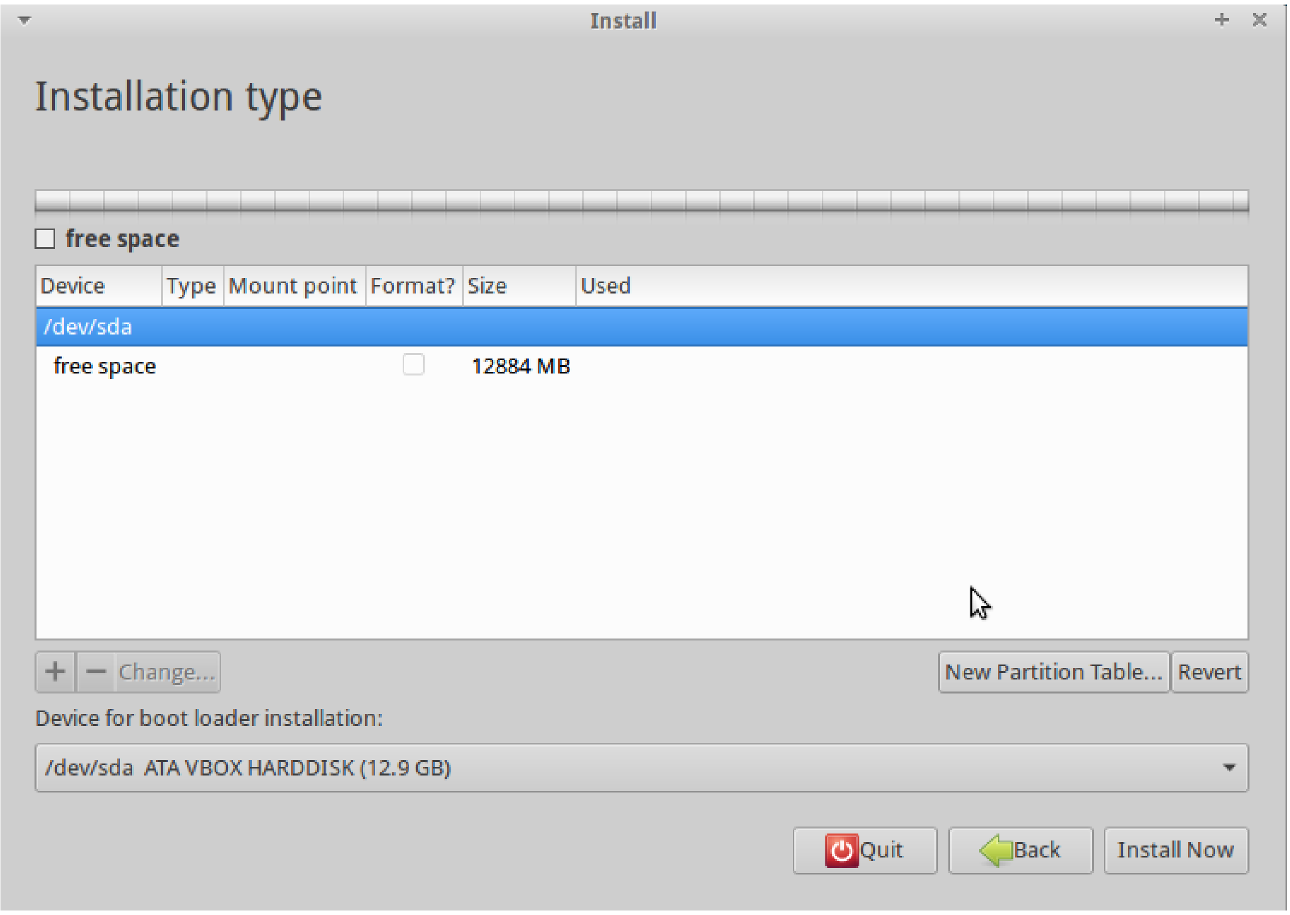
Device	Type	Mount point	Format?	Size	Used
/dev/sda					

Device for boot loader installation:

/dev/sda ATA VBOX HARDDISK (12.9 GB)

Creating a new partition table

Click on New Partition Table to proceed:



Installation type

□ free space

Device	Type	Mount point	Format?	Size	Used
/dev/sda					
free space			□	12884 MB	

Device for boot loader installation:

/dev/sda ATA VBOX HARDDISK (12.9 GB)

This will create a free space as shown in the figure above.

Important: While on your main machine you might see /dev/sda1, /dev/sda2 etc. depending on the number of partitions you have, in that case select the drive by checking its free space and total space to identify which partition You created for Ubuntu and then do the following:

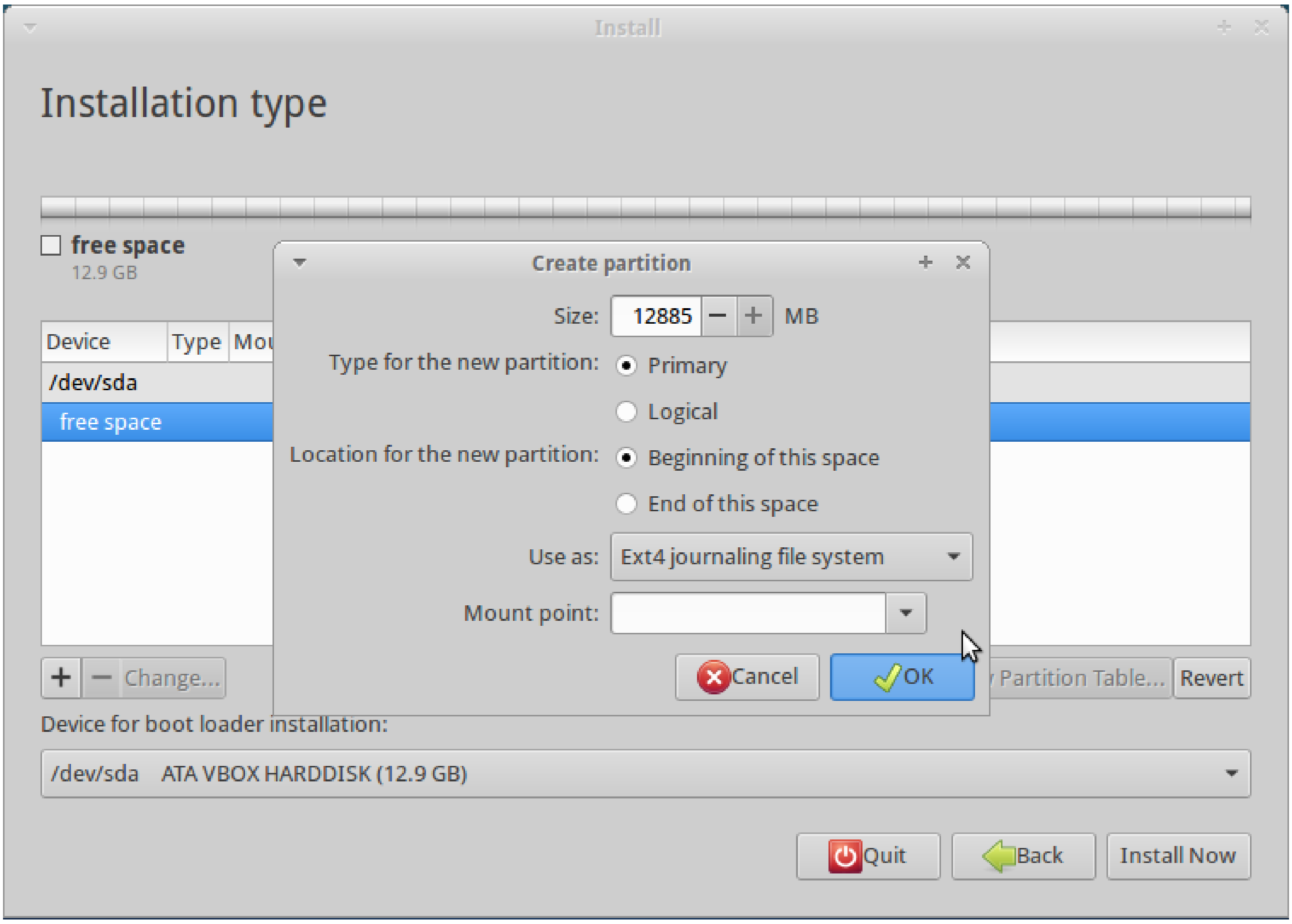
Select the new empty drive you had previously created for Ubuntu, in our case it is the 12884 MB one.

On this free space, we will create partitions required. A default linux installation requires at least one partition. Although, all linux distros recommend making at least two partitions, one to be used as an extended filesystem with the mount point “/” and the other to be used as SWAP area which does not asks for a mount point.

What is a Swap partition?

Allocating a SWAP area can be skipped, but a few hundred megabytes won't hurt and is considered as a good practice. SWAP is used for paging. Memory is loaded with data from disk, or data is written to disk, this process is known as swapping data while only the currently necessary data required is stored in physical memory. While installing linux on main machine, create a 2 GB partition, on USB it can be as low as 300 MB.

Select the device and click “+” on the screen below to get the partition creation table as shown in the next figure:



Installation type

□ free space
12.9 GB

Device	Type	Mou
/dev/sda		
free space		

Size: 12885 MB

Type for the new partition: ● Primary ○ Logical

Location for the new partition: ● Beginning of this space ○ End of this space

Use as: Ext4 journaling file system

Mount point:

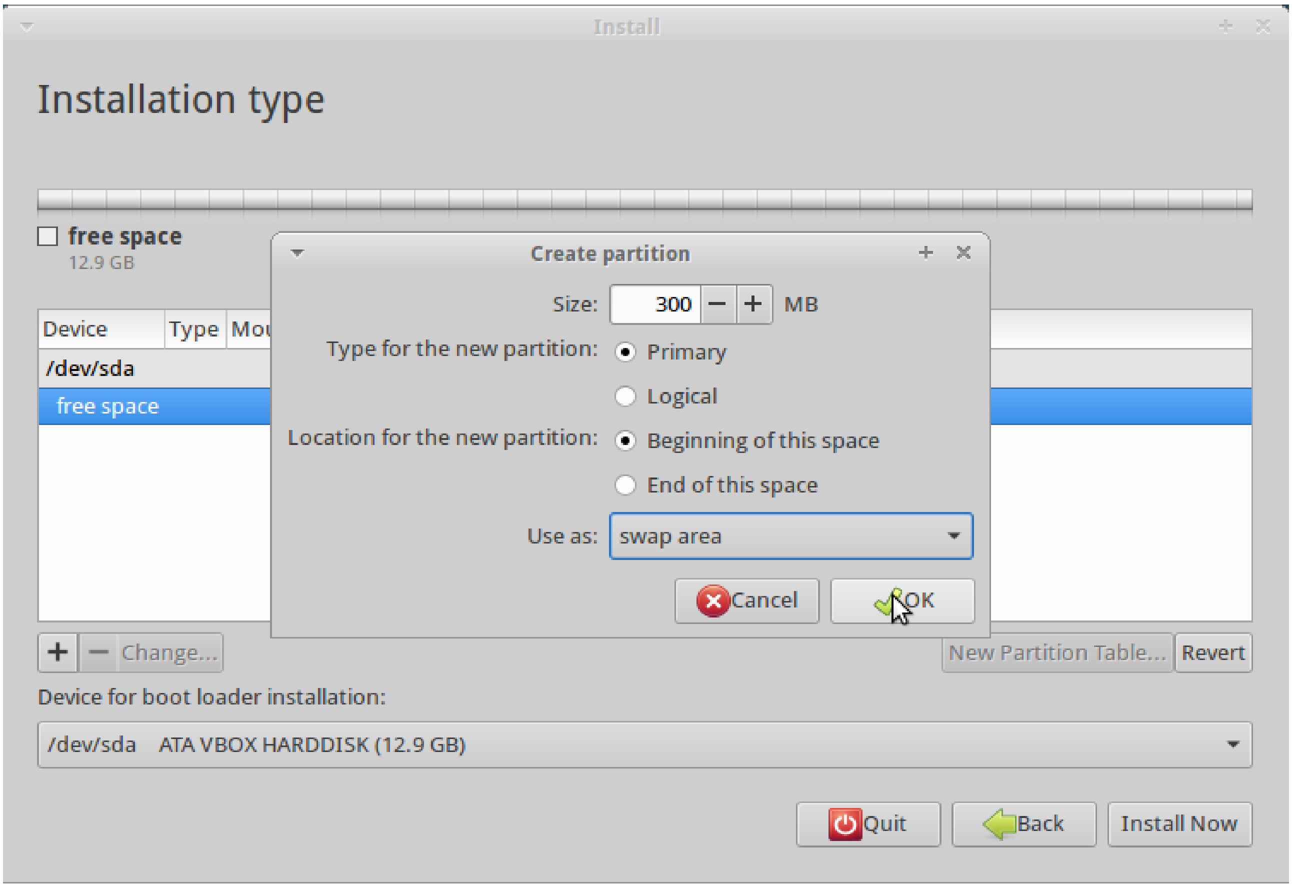
Device for boot loader installation:

/dev/sda ATA VBOX HARDDISK (12.9 GB)

I am going with 300 MB size to be used as SWAP area here.

So now the SWAP area is created and the remaining free space is left.

Let’s then create the main partition that will contain the linux filesystem (“/”). Use it as an Ext4 partition or Ext3.



Installation type

□ free space
12.9 GB

Device	Type	Mou
/dev/sda		
free space		

Create partition

Size: 300 MB

Type for the new partition: ● Primary ○ Logical

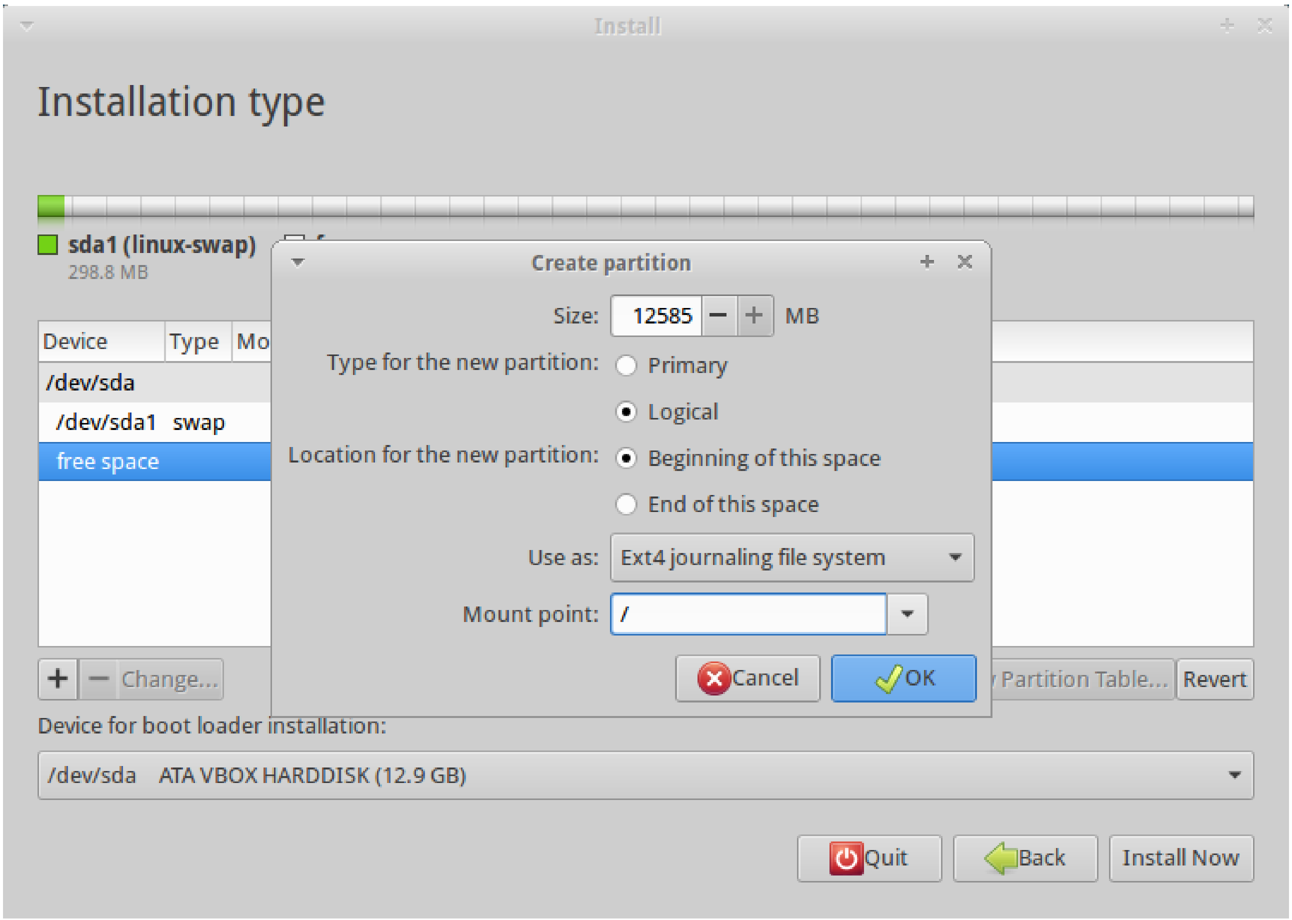
Location for the new partition: ● Beginning of this space ○ End of this space

Use as: swap area

Device for boot loader installation:

/dev/sda ATA VBOX HARDDISK (12.9 GB)

Creating a SWAP Partition



Installation type

🟩 sda1 (linux-swap)
298.8 MB

Device	Type	Mo
/dev/sda		
/dev/sda1	swap	
free space		

Create partition

Size: 12585 MB

Type for the new partition: ○ Primary ● Logical

Location for the new partition: ● Beginning of this space ○ End of this space

Use as: Ext4 journaling file system

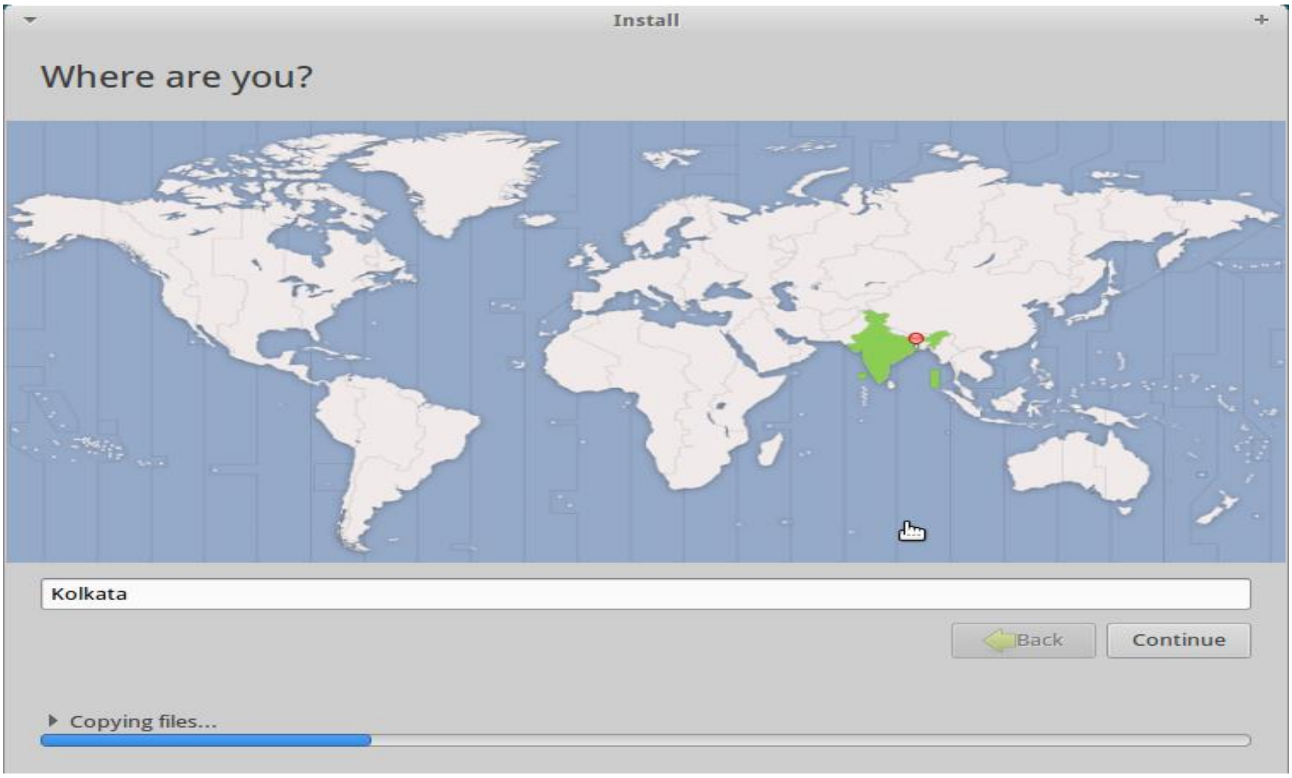
Mount point: /

Device for boot loader installation:

/dev/sda ATA VBOX HARDDISK (12.9 GB)

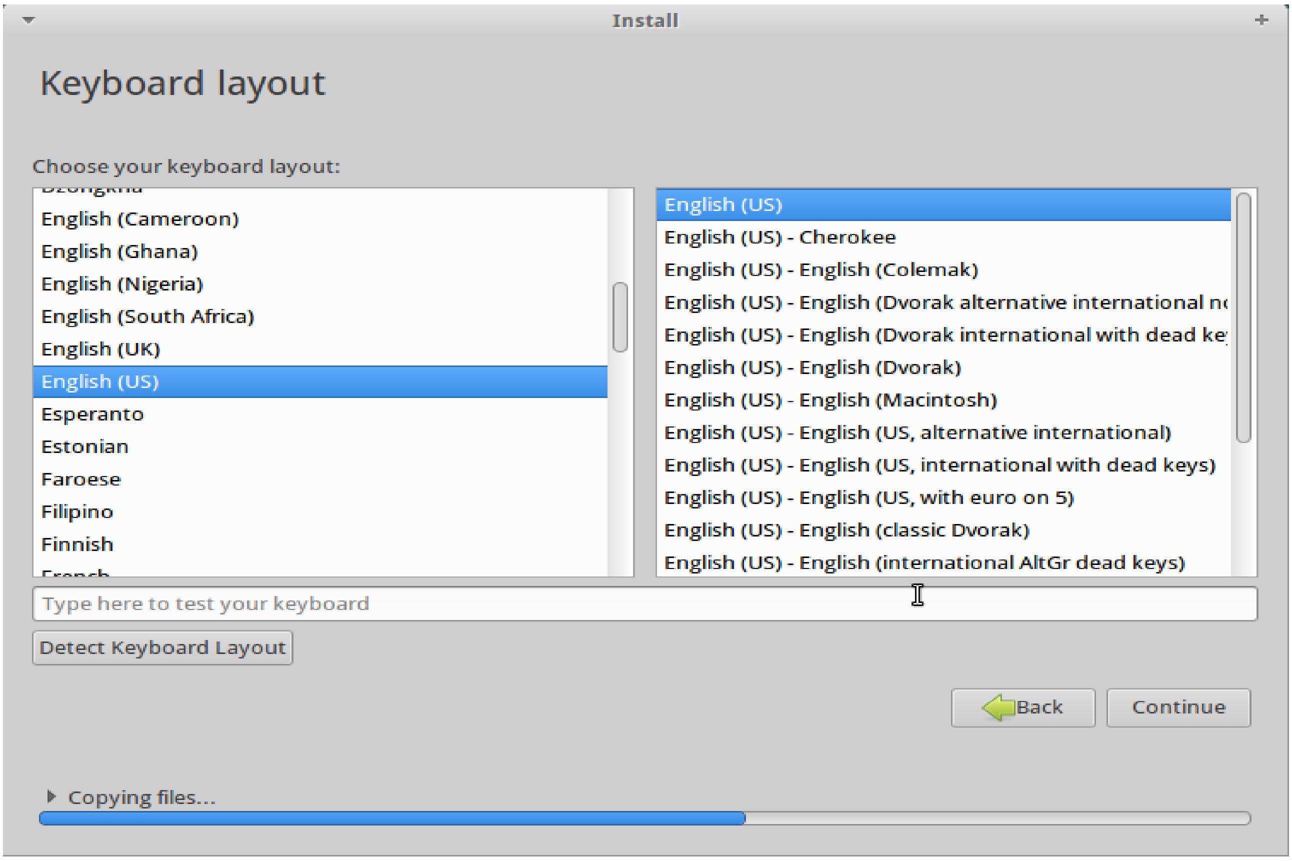
Create an ext4 partition to mount “/”

Click on “Install Now”. The next screen asks for your geographic location.



Kolkata

Set Geo-location



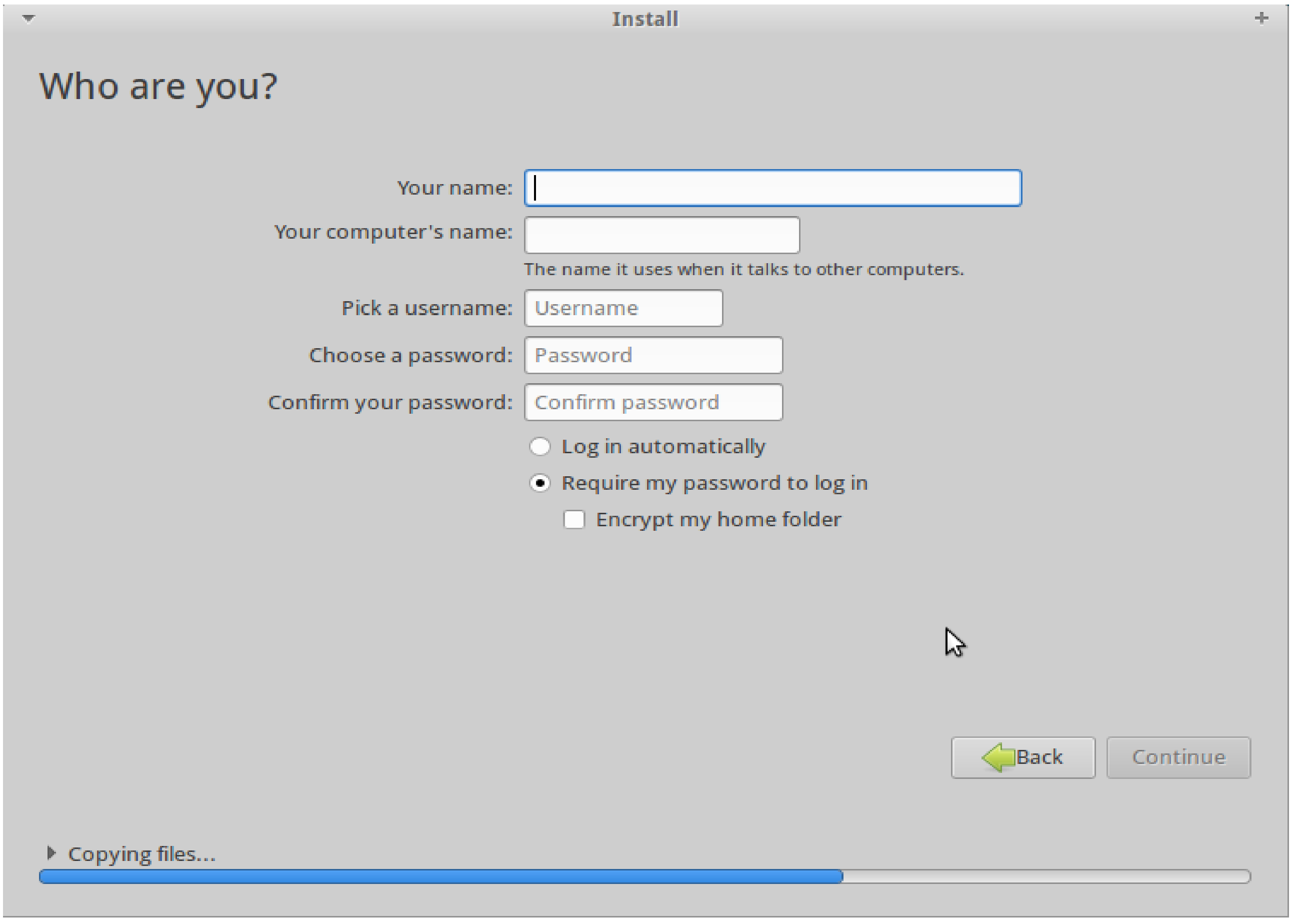
Choose your keyboard layout:

Dzongkha
English (Cameroon)
English (Ghana)
English (Nigeria)
English (South Africa)
English (UK)
English (US)
Esperanto
Estonian
Faroese
Filipino
Finnish
French

English (US)
English (US) - Cherokee
English (US) - English (Colemak)
English (US) - English (Dvorak alternative international nc
English (US) - English (Dvorak international with dead ke
English (US) - English (Dvorak)
English (US) - English (Macintosh)
English (US) - English (US, alternative international)
English (US) - English (US, international with dead keys)
English (US) - English (US, with euro on 5)
English (US) - English (classic Dvorak)
English (US) - English (international AltGr dead keys)

Set Keyboard Layout

Next fill in details about yourself, consider setting up a strong password.



Who are you?

Your name:

Your computer's name:

The name it uses when it talks to other computers.

Pick a username: Username

Choose a password: Password

Confirm your password: Confirm password

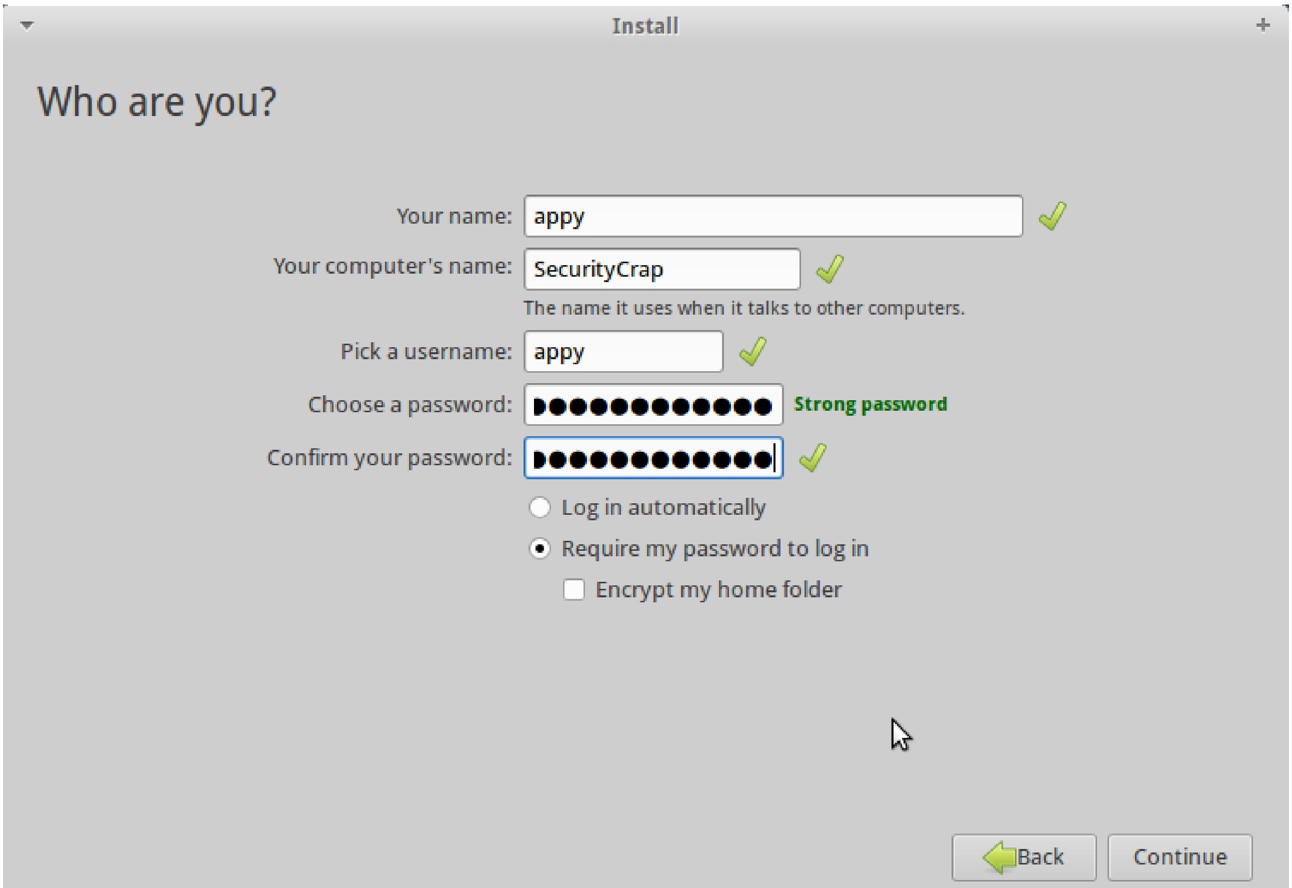
○ Log in automatically

● Require my password to log in

□ Encrypt my home folder

Copying files...

Adding a user



Who are you?

Your name: appy ✓

Your computer's name: SecurityCrap ✓

The name it uses when it talks to other computers.

Pick a username: appy ✓

Choose a password: ●●●●●●●●●●●●● Strong password

Confirm your password: ●●●●●●●●●●●●● ✓

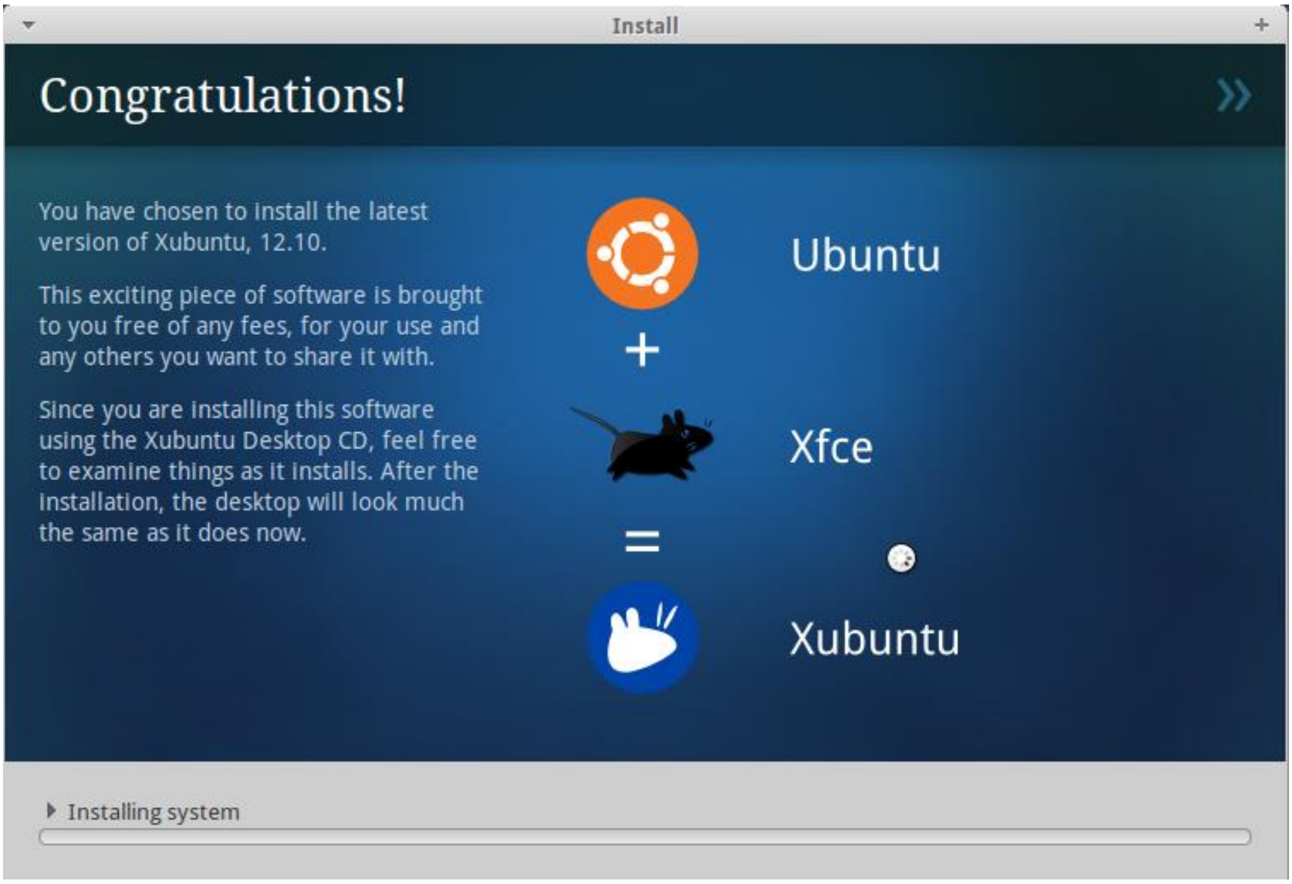
○ Log in automatically

● Require my password to log in

□ Encrypt my home folder

I am going to brag a little about my strong password here ;)

On the “Your computer's name” area, give any name to your machine, any fancy name you wish to give like h4x0r, pwnpad, g33k etc. It’s the hostname or the DNS name of your machine, only the local network machines can see this.

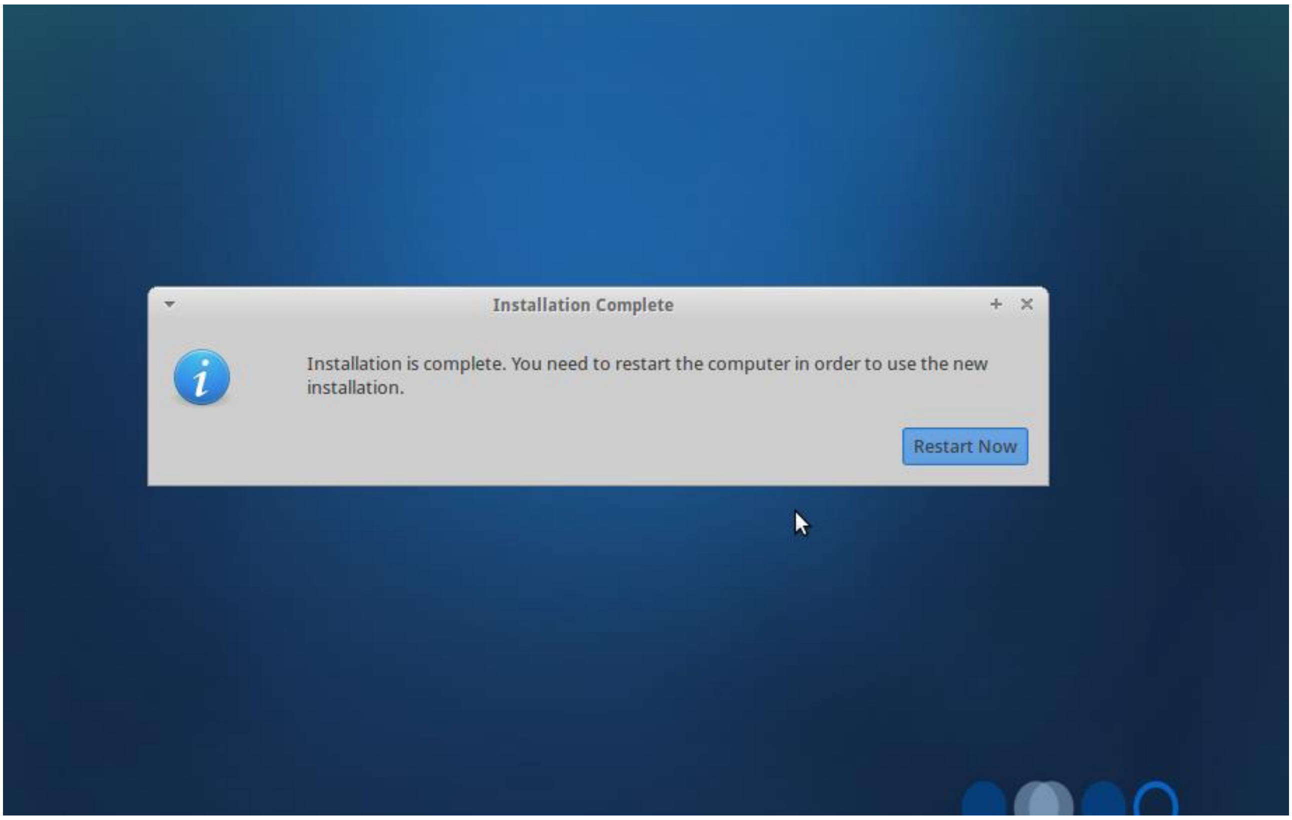


You have chosen to install the latest version of Xubuntu, 12.10.

This exciting piece of software is brought to you free of any fees, for your use and any others you want to share it with.

Since you are installing this software using the Xubuntu Desktop CD, feel free to examine things as it installs. After the installation, the desktop will look much the same as it does now.

It takes only about a matter of 2-5 minutes to finish installation. Cool eh?

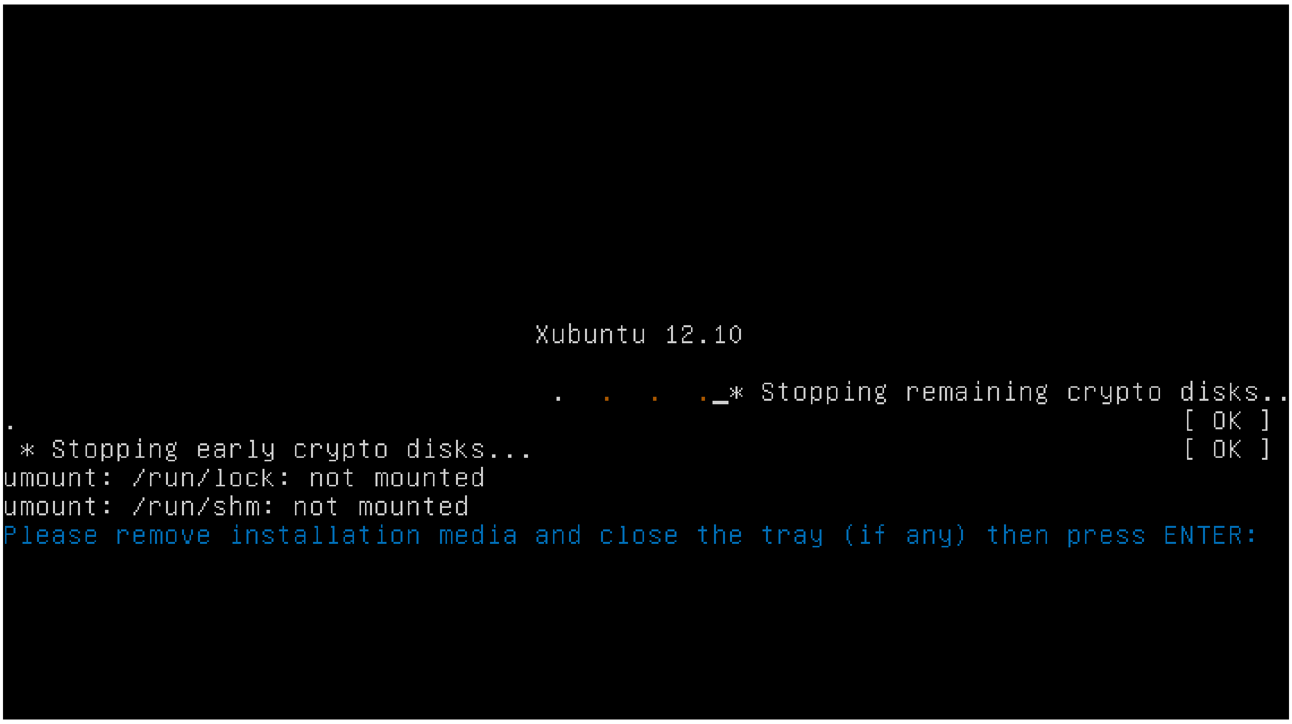


Installation Complete

Installation is complete. You need to restart the computer in order to use the new installation.

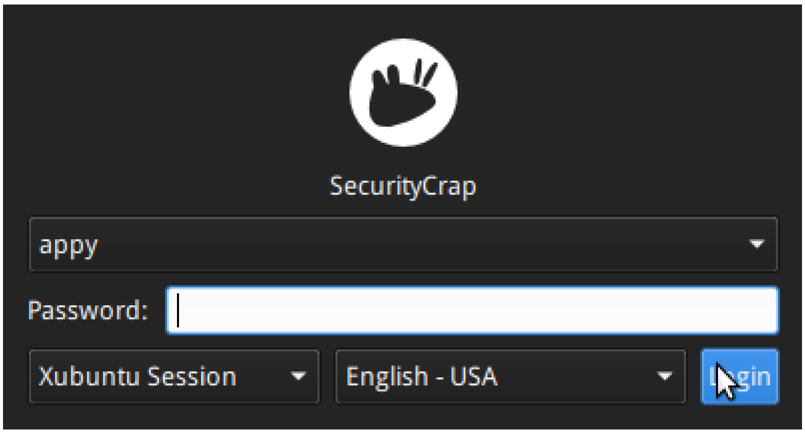
Reboot to finish installation

Click on restart, and then remove the installation media:



```
                              Xubuntu 12.10

                                   .  .  .  ._* Stopping remaining crypto disks..
.                                                                          [ OK ]
 * Stopping early crypto disks...                                          [ OK ]
umount: /run/lock: not mounted
umount: /run/shm: not mounted
Please remove installation media and close the tray (if any) then press ENTER:
```

Virtualbox by default unmounts the installation source, so you just need to click on ENTER. This is the final step, virtual machine will be restarted. You can login now and enjoy the freshly installed Ubuntu.



SecurityCrap

appy

Password:

Xubuntu Session

English - USA

The Login Screen

The virtualbox installation has some display issues in the beginning, the entire screen is not visible in the workspace which is not a big deal. Skip to the “Installing Guest Addition” section in the book to solve that, get back and continue reading.

Extras

Before moving ahead, and before i forget that I mentioned the term runlevel in the beginning while explaining the boot process, for those of you who have managed to come this far reading my book I thought of adding a little more information based on that for you.

Traditionally, there are 7 **runlevels** in Linux:

0	-	halt
1	-	single user mode (also s,S or single)
2	-	multi-user mode without NFS
3	-	full multi-user mode
4	-	undefined (Oops! That makes it 6 now, doesn't it?)
5	-	graphical login
6	-	reboot

Let’s make the count 7 right by adding another runlevel I have found out about, it’s the “emergency” run level. That makes it exactly 7 now.]
Historically, Ubuntu used all the above and by default it was managed by the file /etc/inittab, but now since Ubuntu uses Upstart which has reduced the boot time remarkably, this file no longer exists.

Also, about the runlevels, Ubuntu now handles the following:

0	-	Halt
1	-	Multi-User Text Mode
2	-	Multi-User X Graphical Mode
6	-	Reboot

Runlevels from 3 to 5 can be ignored since Ubuntu treats them all as runlevel 2.

We often shutdown the Linux machine from the terminal by issuing the command: halt or #init 0 and restart the system by issuing the command #init 6 or #reboot.

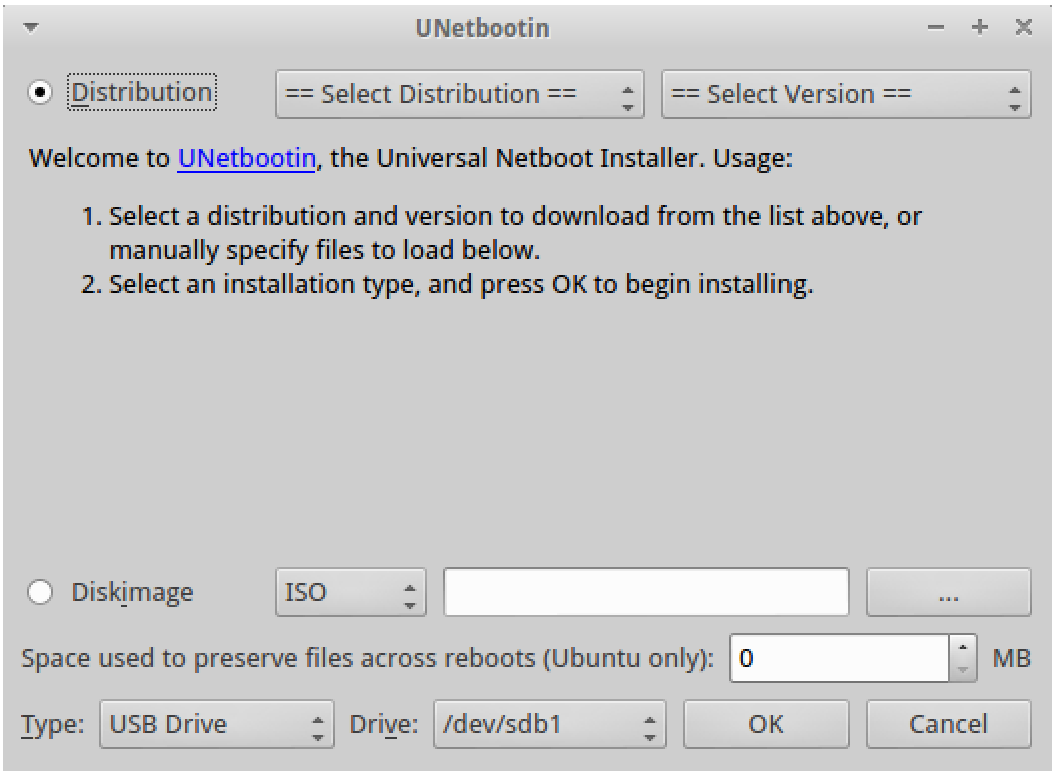
Creating a bootable USB drive

The beauty of Live distros is that you can work on the OS without making any changes to disk, this live distro could be present either in a CD/DVD or a USB. Just like we burn an iso into a DVD, we will do the same on a USB drive using the tool “Unetbootin”. With Unetbootin, we will create a bootable linux USB drive, which can be used to install linux on machines or if you are unsure about installation, then use it in the live-boot mode and work on it without making any changes to the primary disk.

In the default creation, any changes done or files stored while working on the live linux distro will only be temporary, on rebooting those changes and data will be lost. To take care of that, you can allot some persistent storage space in the section just above the confirmation dialog where it says: “Space used to preserve files across reboots”.

In this installation using Unetbootin, we are creating a live distro of linux on the USB drive. This is different from what we did on the main drive or on virtualbox, that was a complete installation and cannot be used to install linux on other machines by live boot.

Below is the snapshot of the Unetbootin application:



● Distribution == Select Distribution == == Select Version ==

Welcome to UNetbootin, the Universal Netboot Installer. Usage:

1. Select a distribution and version to download from the list above, or manually specify files to load below.
2. Select an installation type, and press OK to begin installing.

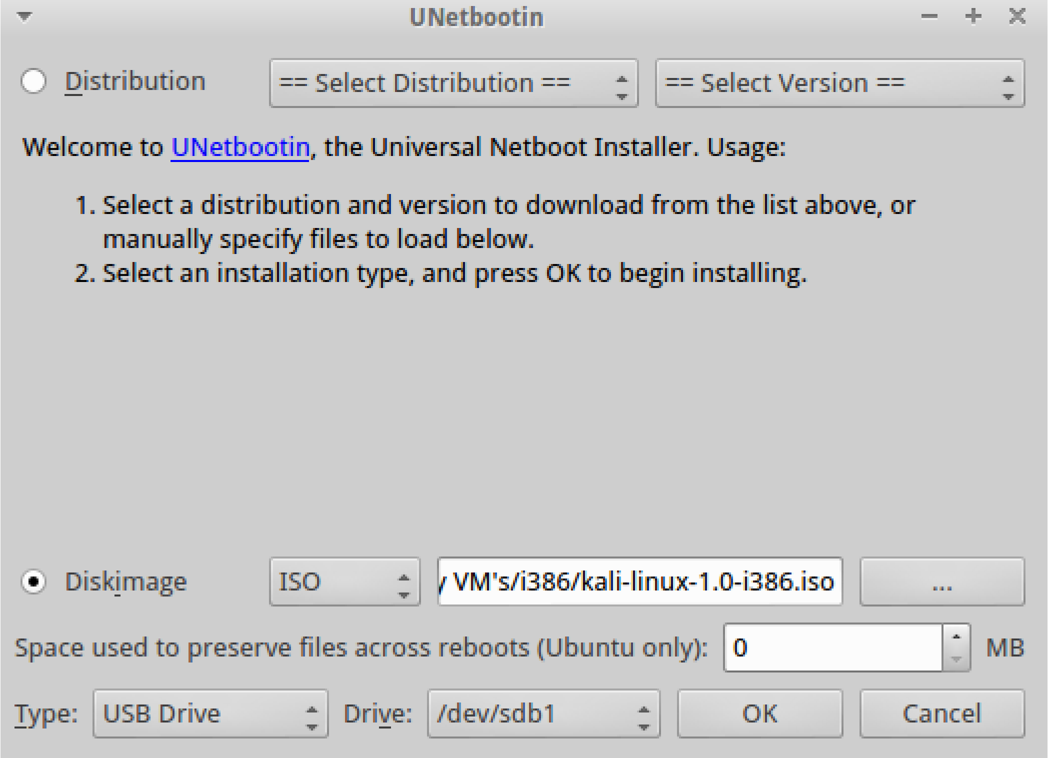
○ Diskimage ISO

Space used to preserve files across reboots (Ubuntu only): 0 MB

Type: USB Drive Drive: /dev/sdb1

Unetbootin

The “Distribution” button is selected by default, the menu list next to it has a list of linux distributions, and choosing one from there will begin downloading the iso from the official website. You can choose that option if you have good download speed and unlimited download plan. My present connection which is not working today would take 4 hours to download a 700 MB file, so the next option below, the “Diskimage” comes to the rescue. With this option you can point Unetbootin to the destination folder where we already have an existing iso file, whew!



○ Distribution == Select Distribution == == Select Version ==

Welcome to UNetbootin, the Universal Netboot Installer. Usage:

1. Select a distribution and version to download from the list above, or manually specify files to load below.
2. Select an installation type, and press OK to begin installing.

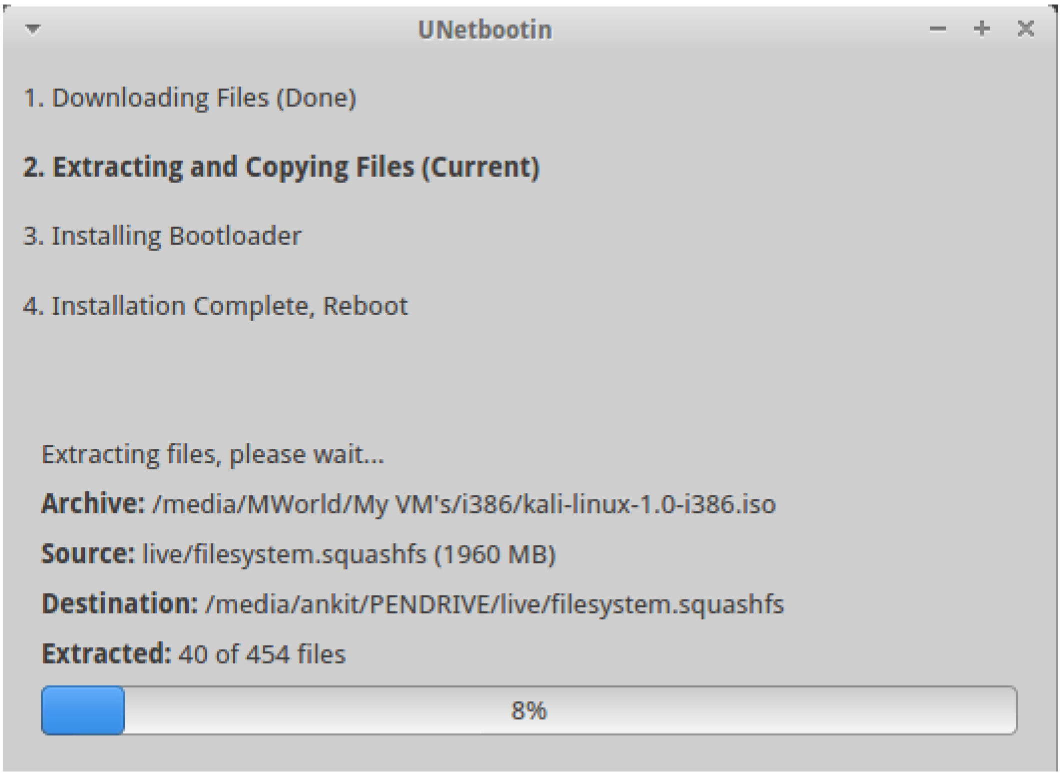
● Diskimage ISO / VM's/i386/kali-linux-1.0-i386.iso

Space used to preserve files across reboots (Ubuntu only): 0 MB

Type: USB Drive Drive: /dev/sdb1

Setting diskimage & USB drive

I have selected a different linux distro here. Just below it you can see Unetbootin has already found the USB drive attached to the machine → /dev/sdb1 here. That's it, the application is good to go for the installation.



1. Downloading Files (Done)

2. Extracting and Copying Files (Current)

3. Installing Bootloader

4. Installation Complete, Reboot

Extracting files, please wait...

Archive: /media/MWorld/My VM's/i386/kali-linux-1.0-i386.iso

Source: live/filesystem.squashfs (1960 MB)

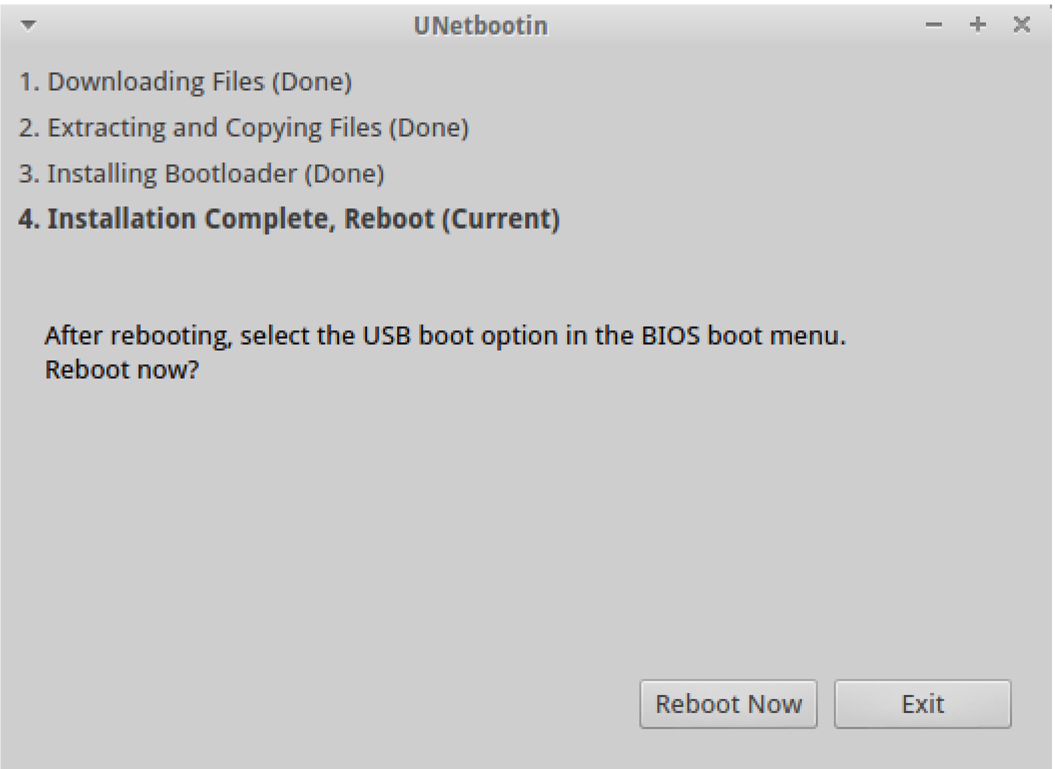
Destination: /media/ankit/PENDRIVE/live/filesystem.squashfs

Extracted: 40 of 454 files

8%

Installation

It takes less than 5 minutes to finish installation, you don't necessarily need to reboot the machine on completion as prompted.



1. Downloading Files (Done)
2. Extracting and Copying Files (Done)
3. Installing Bootloader (Done)
4. Installation Complete, Reboot (Current)

After rebooting, select the USB boot option in the BIOS boot menu.
Reboot now?

And there you have your bootable linux pen drive. Plug it on any PC, and boot it from the USB drive. Even if you don't wish to make permanent changes to the hard drive by installing linux, you can still enjoy working on it with the “Try without installing” feature.

It’s kind of a safe practice while working on public or unknown devices where you suspect presence of keyloggers or viruses. Also, having your own pentesting OS on a portable device sometimes comes in handy.

Over the years I have always carried some flavour of linux on my pen drive, and have found the ultimate answer to the question of “What do you do for fun?”,
“I install linux on things”. ;)

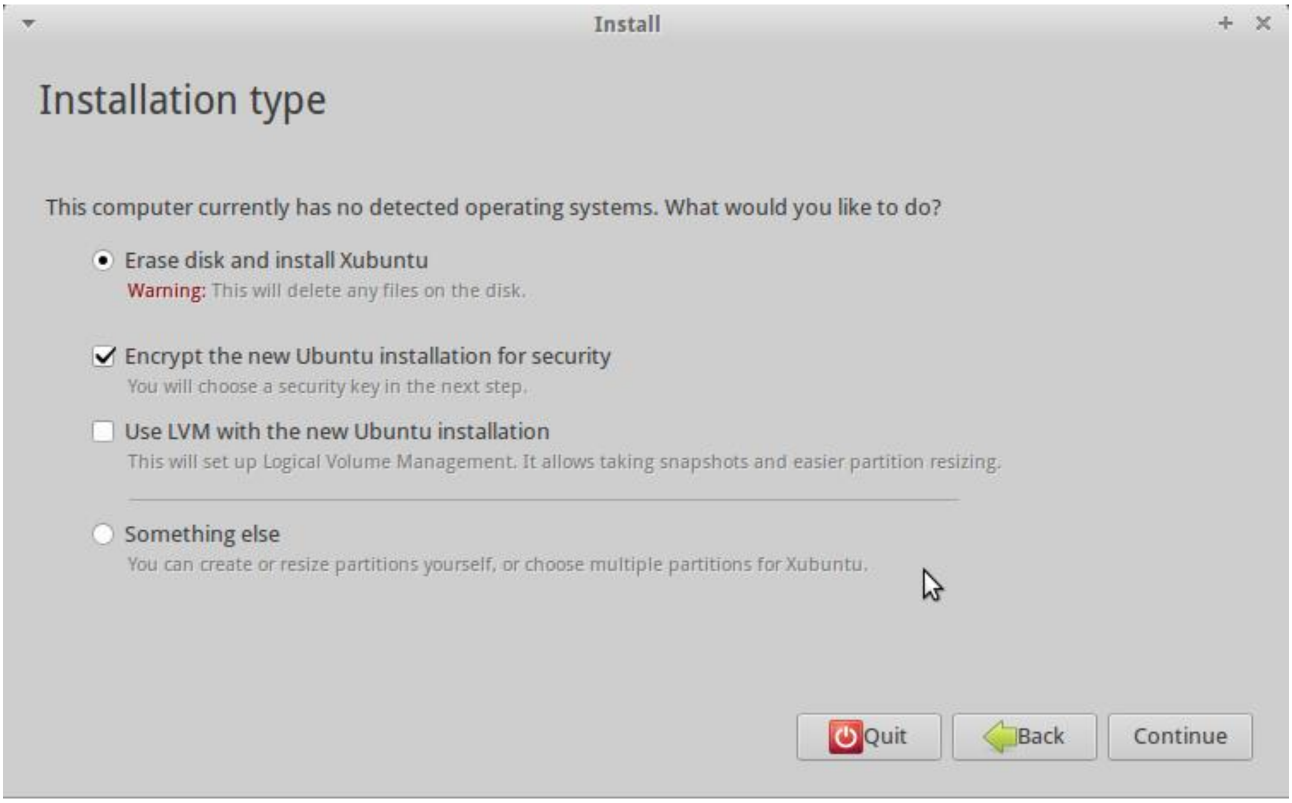
Full-disk Encrypted Linux Installation

During the installation, while selecting a disk to do the installation on, we get an option to encrypt the disk first. Again, make sure there is nothing to lose in the disk drive before selecting this option, because if you choose to perform this kind of installation, entire hard disk will be wiped clean and then encrypted.

This One-Click Full-Disk encryption is one amazing feature, now every time you boot-up your OS, it will ask for the passkey to decrypt the filesystem to continue booting. Without this passkey, you will not be presented with the login screen.

Benefit, you ask? Now, even if someone gets physical access to your machine, or takes your hard drive, he won't be able to read the files saved in the disk. Without the login credentials it was possible for him to read the files, not it's not. So starting from the screen where we are asked to select the installation type:

Click on the check box: "Encrypt the new Ubuntu installation".



Installation type

This computer currently has no detected operating systems. What would you like to do?

● Erase disk and install Xubuntu
Warning: This will delete any files on the disk.

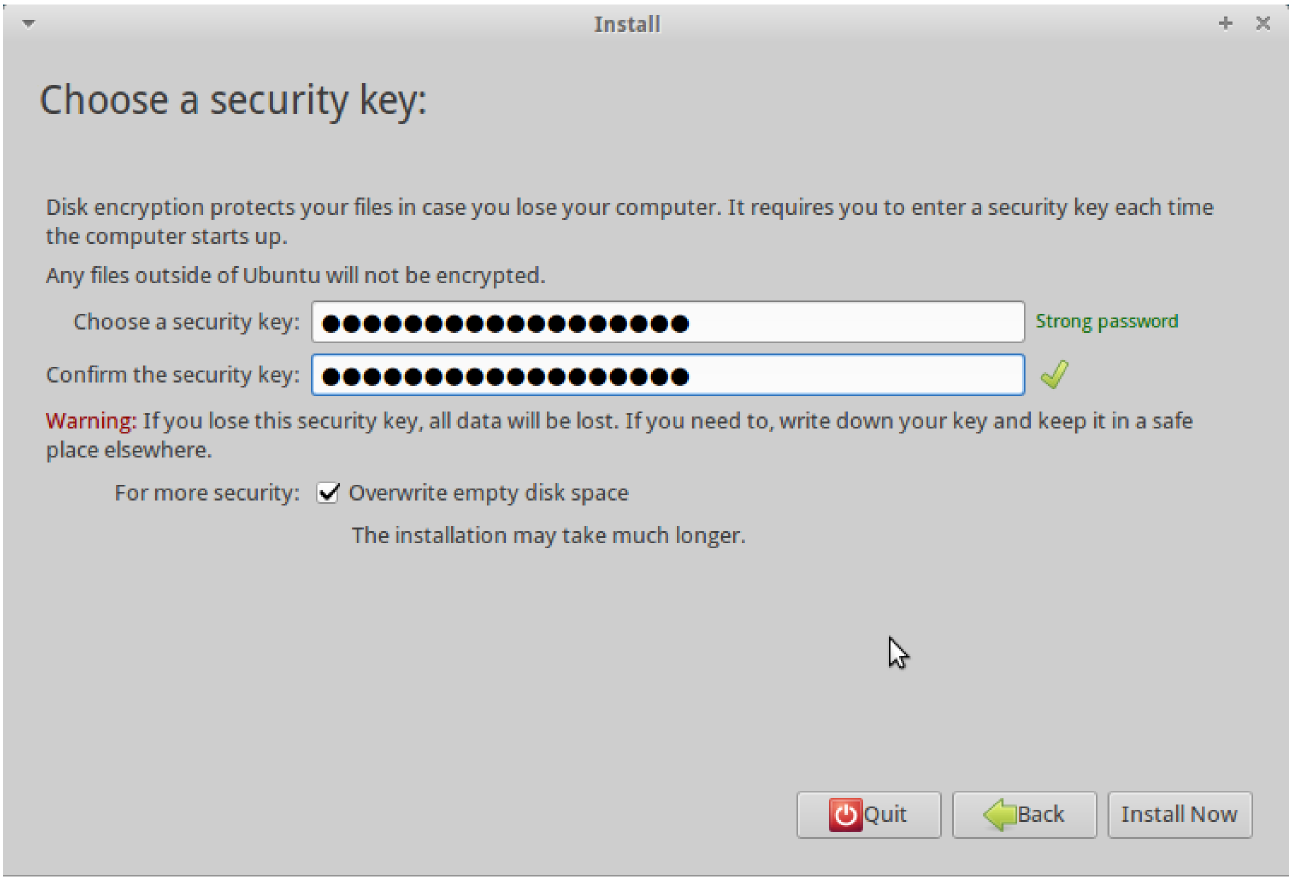
☑ Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

□ Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

○ Something else
You can create or resize partitions yourself, or choose multiple partitions for Xubuntu.

Choosing the installation type

And choose a strong password in the next step:



Choose a security key:

Disk encryption protects your files in case you lose your computer. It requires you to enter a security key each time the computer starts up.

Any files outside of Ubuntu will not be encrypted.

Choose a security key: ●●●●●●●●●●●●●●●●●●● Strong password

Confirm the security key: ●●●●●●●●●●●●●●●●●●● ✓

Warning: If you lose this security key, all data will be lost. If you need to, write down your key and keep it in a safe place elsewhere.

For more security: ☑ Overwrite empty disk space

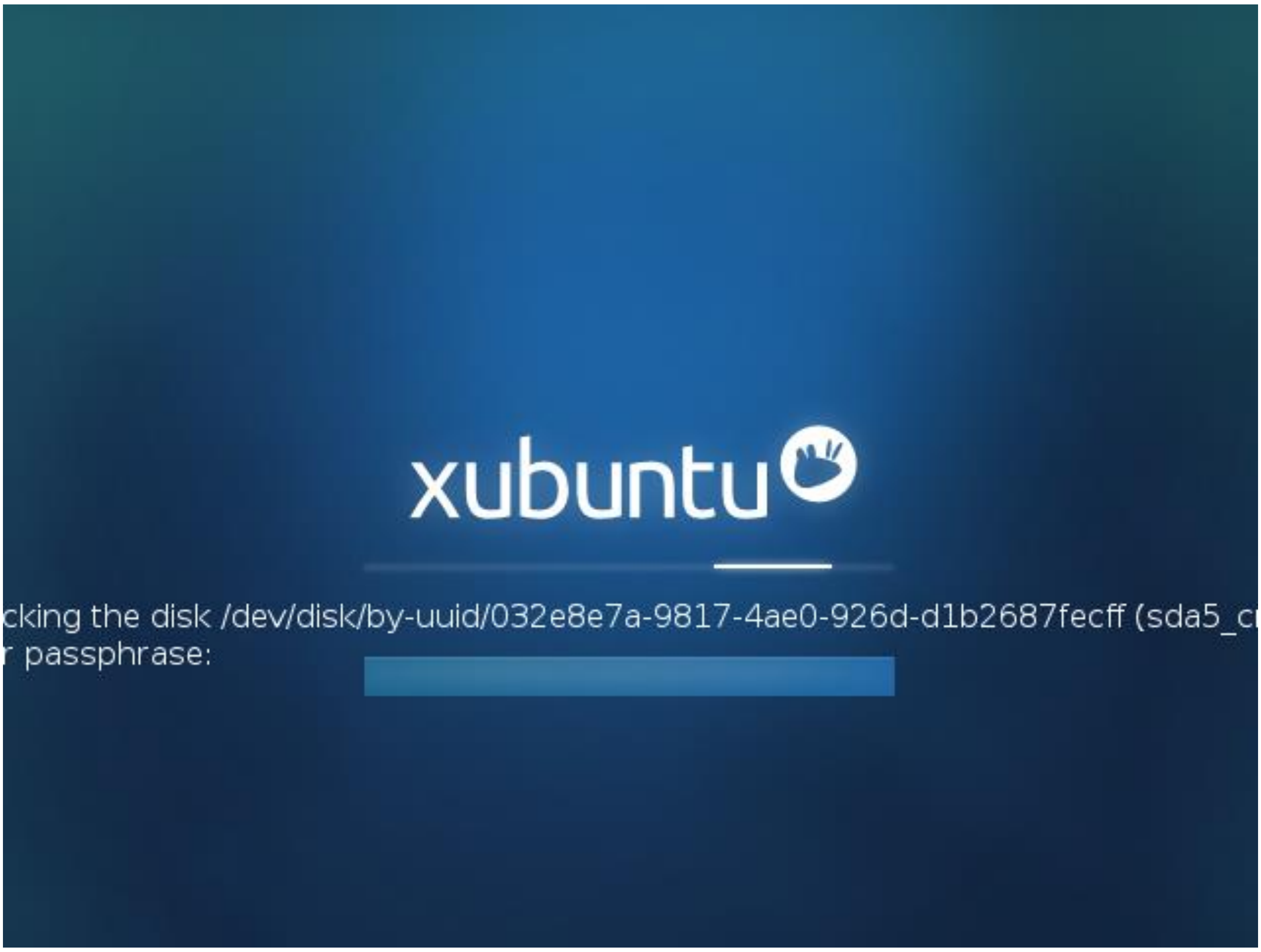
The installation may take much longer.

Setting a strong disk encryption password

Yet again, you can see I have chosen a “Strong Password”, I’d recommend choosing at least a 10 character password with at least: 1 Upper-Case letter, a few lower-case letters, 4 Numbers and 2 special characters.
Rest of the steps are same as already shown in the previous installation. Or an even better combination would be a phrase with spaces.

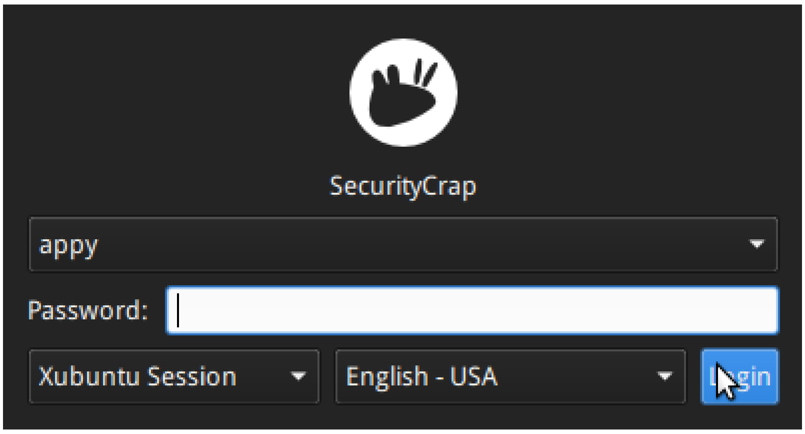
Again, click enter after removing the installation media and reboot. The Xubuntu interface will halt and you will be asked to enter the security key that was used to encrypt the entire disk, as shown in the figure on the next page.

There is no workaround if you forget this passphrase. Without it, you will not be presented with the login screen and you will have to perform the installation all over again. So, you can see how important it is to remember this password.



```
cking the disk /dev/disk/by-uuid/032e8e7a-9817-4ae0-926d-d1b2687fecff (sda5_c
r passphrase:
```

Enter the passphrase, to continue logging in and using Ubuntu.



SecurityCrap

appy

Password:

Xubuntu Session

English - USA

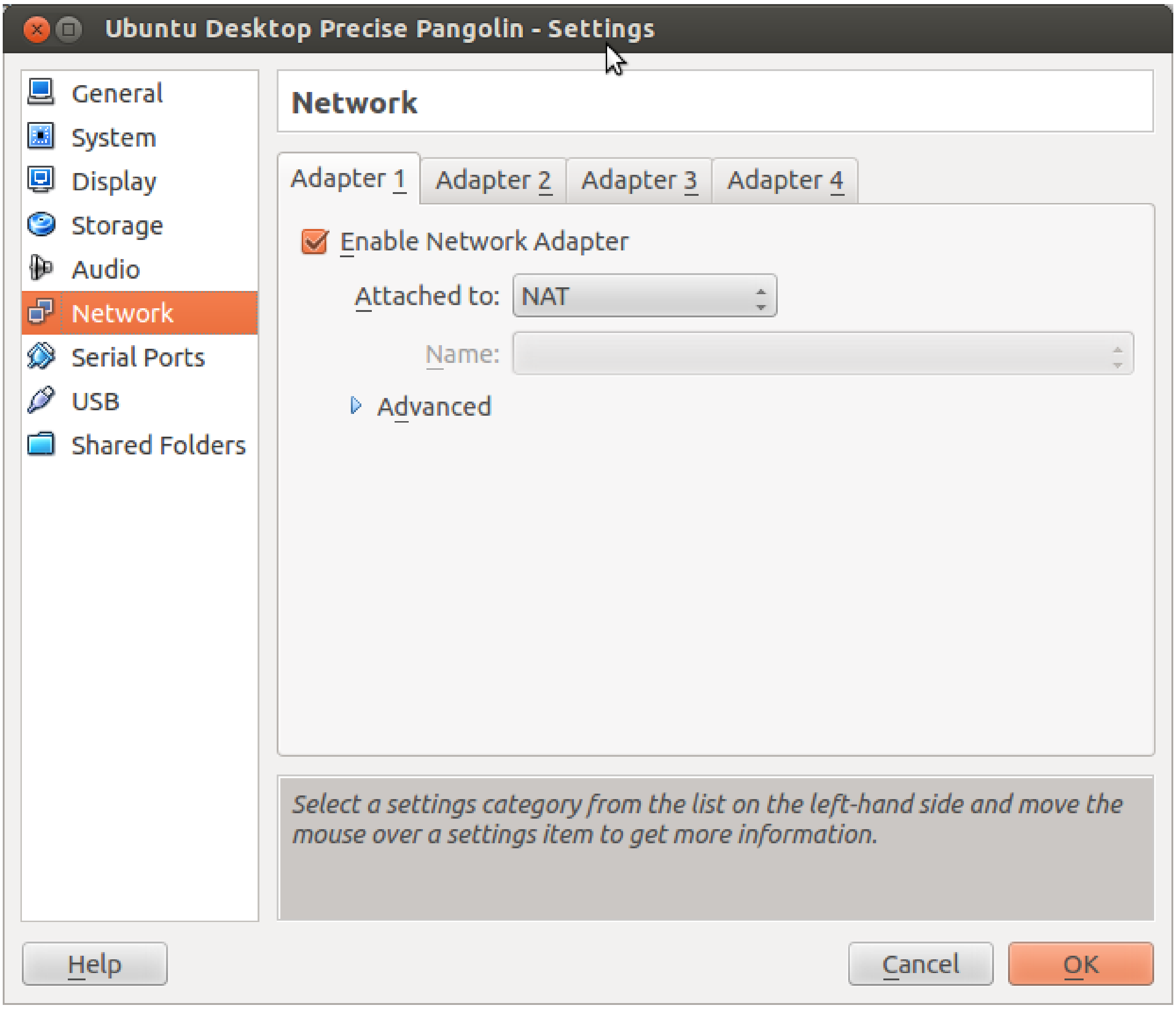
Virtual Box

Virtual Box is an Open Source Application that provides amazing Virtualization features. You can run multiple Operating systems under one Operating System and do a lot of fancy stuffs, and it's available for all platforms: Windows, Linux and OSX. With the different view modes, it becomes fairly simple to work simultaneously on more than one Operating System.

Let's check out some of the cool features of virtualbox:

Network Adapters

On a virtual machine, we can have four active virtual adapters. The network adapters can be selected from the settings button on the top of the main VirtualBox windows. Select the Network tab from the settings:



Ubuntu Desktop Precise Pangolin - Settings

Network

☑ Enable Network Adapter

Attached to: NAT

Name:

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

The four adapters that can be switched on at the same time.

There are 5 Network Adapters supported in Virtual Box:

NAT - (Network Address Translation)

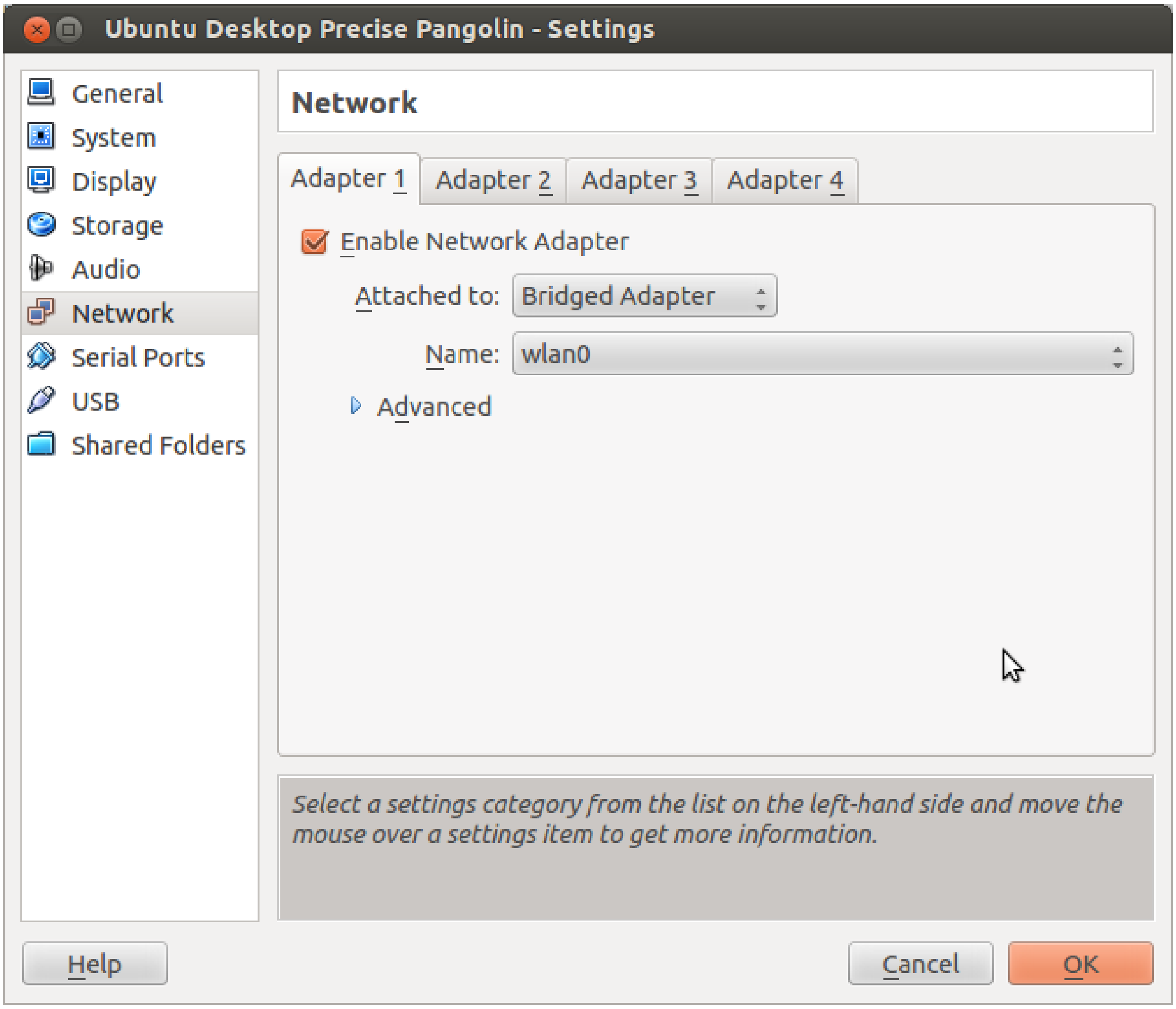
This adapter is set by default and will connect to the internet in the virtual machine if the host machine has a working connection. It works exactly like NAT in modern ADSL modems/routers. For quite some time the only feature I believed it had to offer was to connect to the internet, which it does but there is more to it. Just the way your internal IP address is not visible to the outside network, the virtual machine's IP isn't. Instead, packets going from this virtual machine will appear to be coming from the host machine's IP address. Extremely useful while doing pentests from multiple distros simultaneously from a single IP. By default an IP of the 10 series is assigned.

Bridged Adapter

In this mode, the virtual machine connects to the network using the real wlan or lan interface. Upon selecting this adapter an additional option will appear just below it asking You to select one interface, depends on the machine, it sometimes just asks You to choose between "wlan" or "lan", sometimes it shows the name of the adapter namely Intel Wireless or something like that. Choosing the appropriate option will let virtual machine connect to the network. For instance, if you have connected to the internet using a WLAN, then select the wlan0 interface, upon doing that the modem will consider this virtual machine to be a real machine and it will be assigned an IP address of the same series as the host machine and other machines in the network. This virtual machine can now be reached via all other machines in the local network. All traffic goes through a physical interface of the host system.

Internal Network

This mode allows two or more virtual machines to interact with each other without the interference of the host machine. Only the VMs can interact with each other, not even with the host machine.



☑ Enable Network Adapter

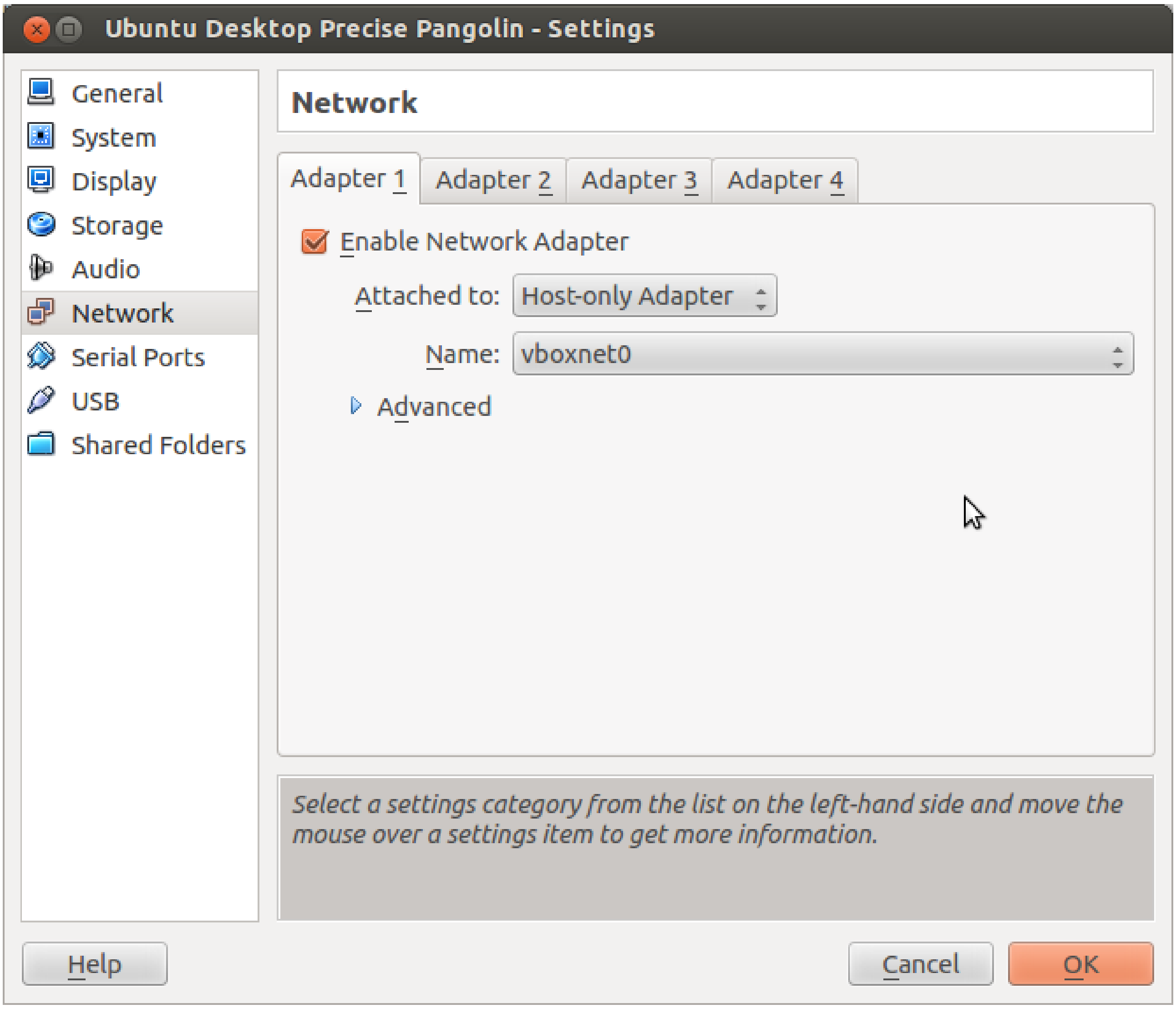
Attached to: Bridged Adapter

Name: wlan0

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

Host-Only Adapter

In this mode, the virtual machines can interact with each other and with the host machine. There is an interference of host machine. While selecting this adapter, there is a new virtual interface that is created in the host machine, vboxnet. In the recent releases of virtualbox that interface is not created automatically. So in order to use host-only adapter, first click on “Files” from the main virtualbox window and select “Preferences”. There will be a network tab, in that window you will see “Host-Only Adapters:” followed by an empty box. On the right side of that box is a + button, click on that to create the first virtual interface, i.e. Vboxnet0. You can add as many interfaces as you want. Now, go to the network setting of the individual machine and opt for host-only adapter to see if you are prompted with vboxnet. The moment you turn the VM on there will be a new virtual adapter created. Check this by the command “ifconfig” in your host machine. The default IP address of the host machine's vboxnet interface is 192.168.56.1 for vboxnet0, for vboxnet1 it



Ubuntu Desktop Precise Pangolin - Settings

Network

☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: vboxnet0

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

is 192.168.57.1 and similarly for the rest. For the first virtual machine using host-only adapter, the default IP is 192.168.56.101. So all the machines using the same vboxnet adapter will be able to interact with each other and the host machine.

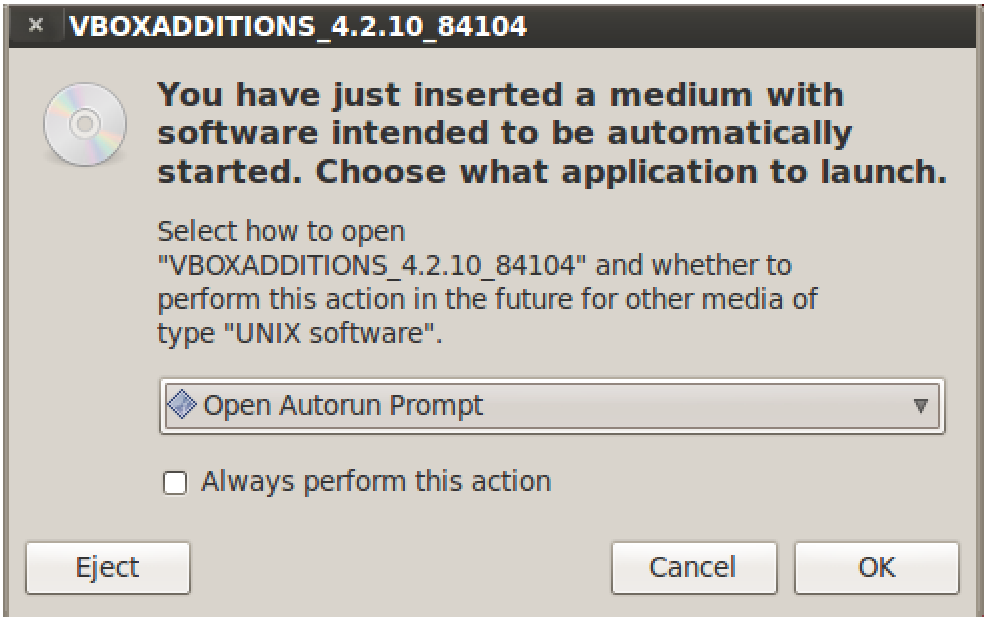
- Generic Driver

This adapter offers an open plugin architecture for arbitrary and separately distributable virtual network implementations.

Installing Guest Additions

It's highly recommended to install **Guest Additions** after you install your first OS using Virtual Box. It gives a new look to your virtual OS.
To do that, you just have to click on the "Devices" option on the top and a list will open up. In the bottom of that list there is the option "Install Guest Additions".

Clicking on that will open a window inside the virtual machine asking permission for auto-run.



× VBOXADDITIONS_4.2.10_84104

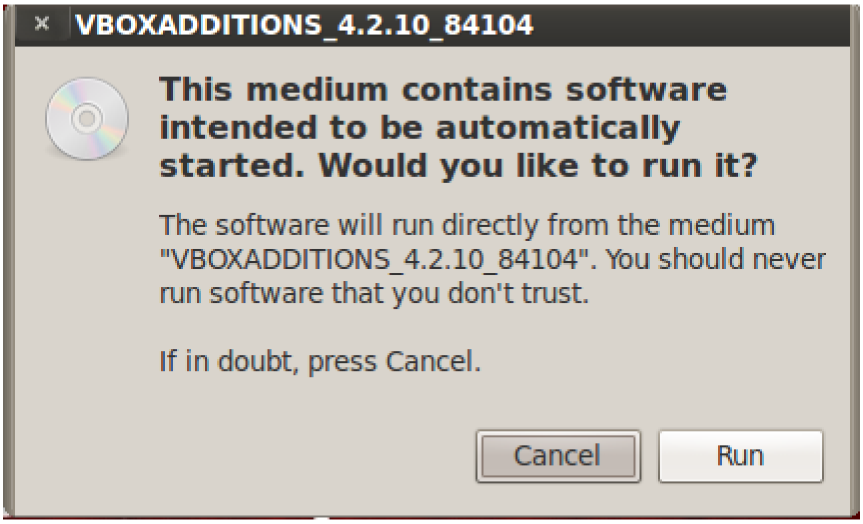
You have just inserted a medium with software intended to be automatically started. Choose what application to launch.

Select how to open "VBOXADDITIONS_4.2.10_84104" and whether to perform this action in the future for other media of type "UNIX software".

Open Autorun Prompt ▼

□ Always perform this action

And then the next window asking for the final execution permission:



× VBOXADDITIONS_4.2.10_84104

This medium contains software intended to be automatically started. Would you like to run it?

The software will run directly from the medium "VBOXADDITIONS_4.2.10_84104". You should never run software that you don't trust.

If in doubt, press Cancel.

It will begin installation and show the step by step procedure.
Sometimes it fails due to the absence of compatible linux-headers or absence of dkms, but on the newer versions of virtualbox I have never encountered that problem. But it was still worth mentioning. It’s no big deal, you can install dkms and linux-headers are now compatible with the Guest Addition modules. So no sweat.
Another problem could be a non-privileged user.

In that case, switch to the root user in terminal, and navigate to '/media/VBOXADDITIONS_<version no>' and then execute the VboxLinuxAdditions.run script.
Here is the output:

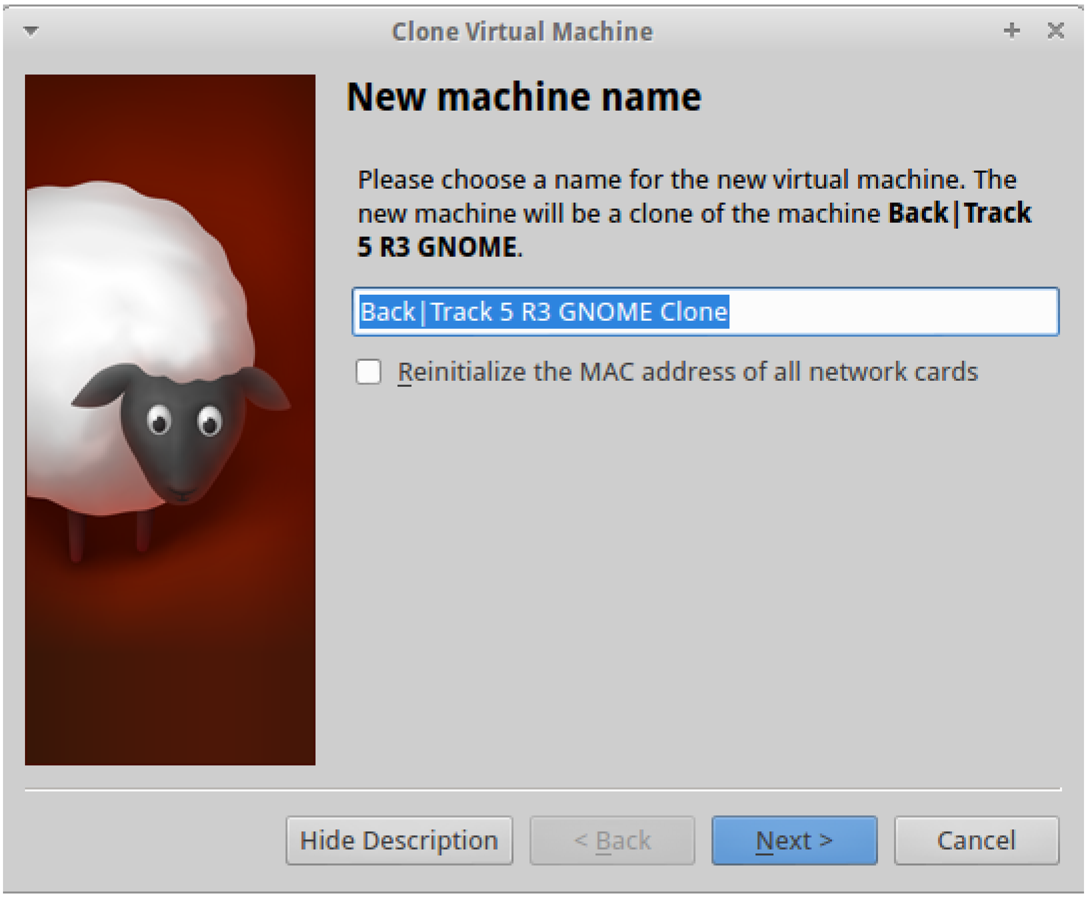
```
Verifying archive integrity... All good.
Uncompressing VirtualBox 4.2.10 Guest Additions for Linux..........
VirtualBox Guest Additions installer
tar: Record size = 8 blocks
Removing existing VirtualBox non-DKMS kernel modules ...done.
Building the VirtualBox Guest Additions kernel modules
Building the main Guest Additions module ...done.
Building the shared folder support module ...done.
Building the OpenGL support module ...done.
Doing non-kernel setup of the Guest Additions ...done.
You should restart your guest to make sure the new modules are actually used

Installing the Window System drivers
Installing X.Org Server 1.7 modules ...done.
Setting up the Window System to use the Guest Additions ...done.
You may need to restart the hal service and the Window System (or just restart
the guest system) to enable the Guest Additions.

Installing graphics libraries and desktop services components ...done.
```

After successful installation of Guest Additions You will have to restart Your VM once. Now if you don't see any change in the VM then just double click on the title bar twice to reduce the screen size, maximize it again and now you will see the VM in its full size inside the VirtualBox. It will now adjust to any size of the VirtualBox window you choose. No need to scroll anymore. Beautiful isn't it?

There is an amazing feature in the "View" bar on the top, the Seamless mode. Select that and now both the host & virtual machine will blend like one. It's like the VM which was previously running inside a Box has now popped out of the box and you are using two OS at the same time. Not just two, you can have three or more OSes running at the same time using the seamless mode.

Cloning

An already installed VM can be cloned as a backup. Two types of cloning are available: Full clone & Linked Clone.

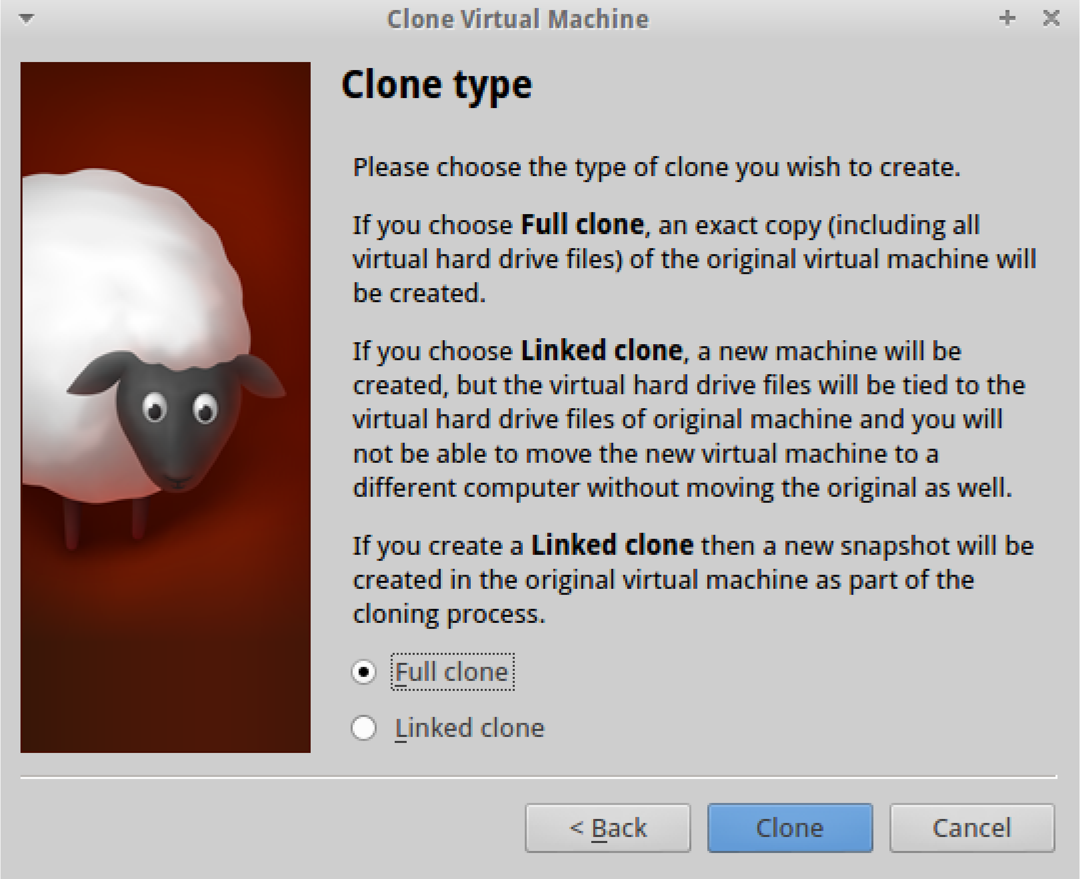


New machine name

Please choose a name for the new virtual machine. The new machine will be a clone of the machine **Back|Track 5 R3 GNOME**.

Back|Track 5 R3 GNOME Clone

□ Reinitialize the MAC address of all network cards



Clone type

Please choose the type of clone you wish to create.

If you choose **Full clone**, an exact copy (including all virtual hard drive files) of the original virtual machine will be created.

If you choose **Linked clone**, a new machine will be created, but the virtual hard drive files will be tied to the virtual hard drive files of original machine and you will not be able to move the new virtual machine to a different computer without moving the original as well.

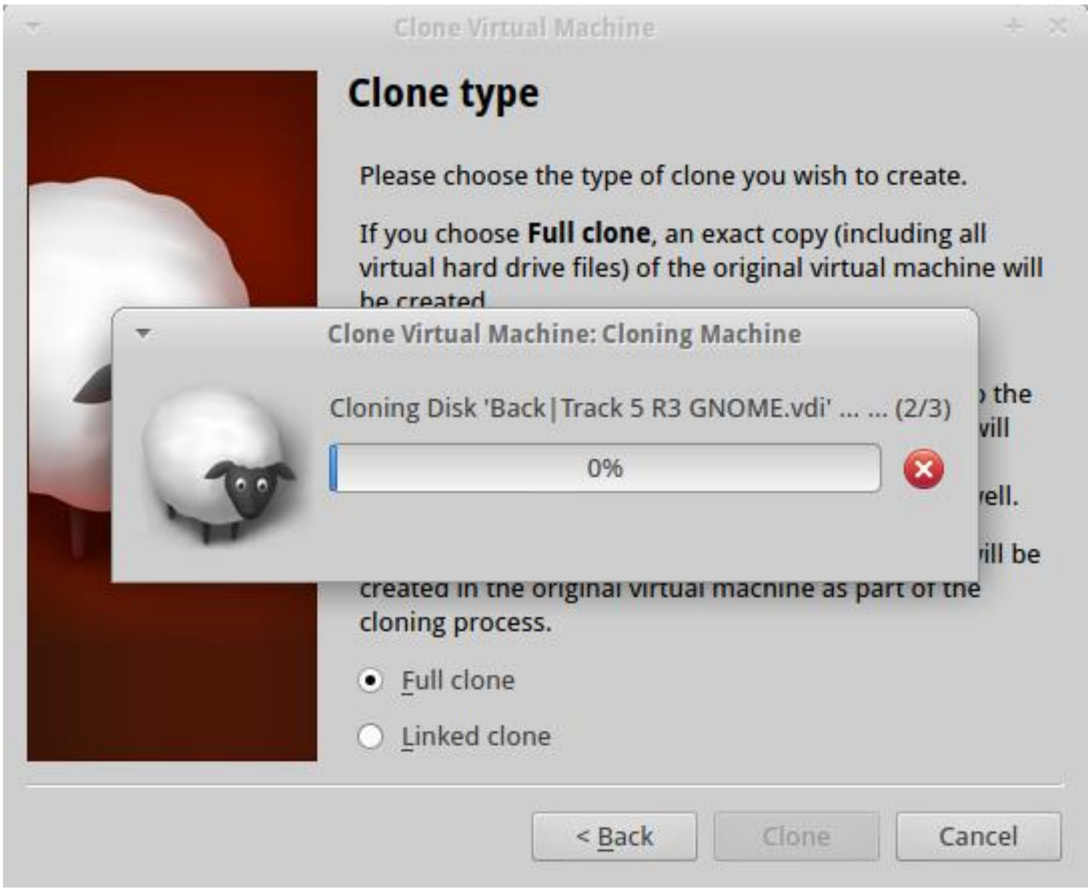
If you create a **Linked clone** then a new snapshot will be created in the original virtual machine as part of the cloning process.

● Full clone

○ Linked clone

Cloning types, with explanation

And the cloning begins:



Clone type

Please choose the type of clone you wish to create.

If you choose **Full clone**, an exact copy (including all virtual hard drive files) of the original virtual machine will be created.

Clone Virtual Machine: Cloning Machine

Cloning Disk 'Back|Track 5 R3 GNOME.vdi' ... ... (2/3)

0%

the
vill

ell.

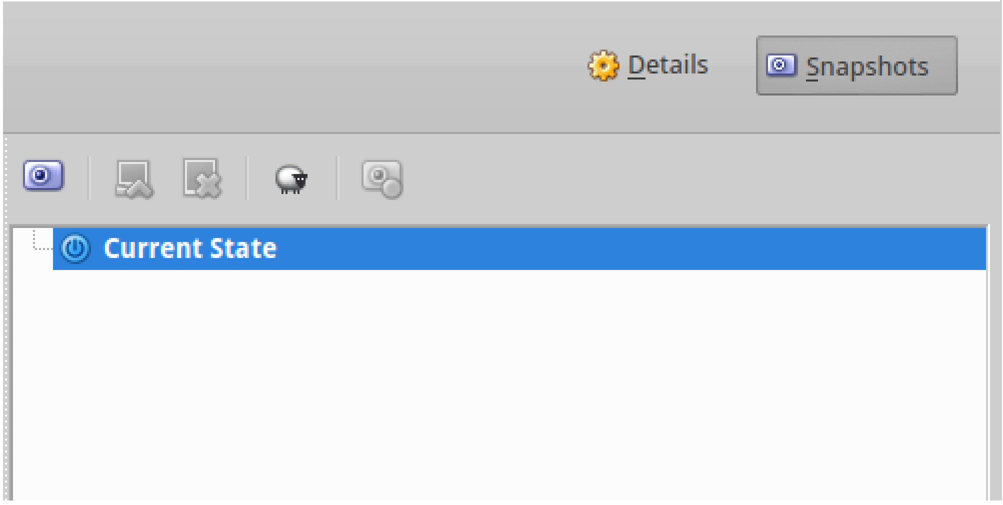
ill be
created in the original virtual machine as part of the cloning process.

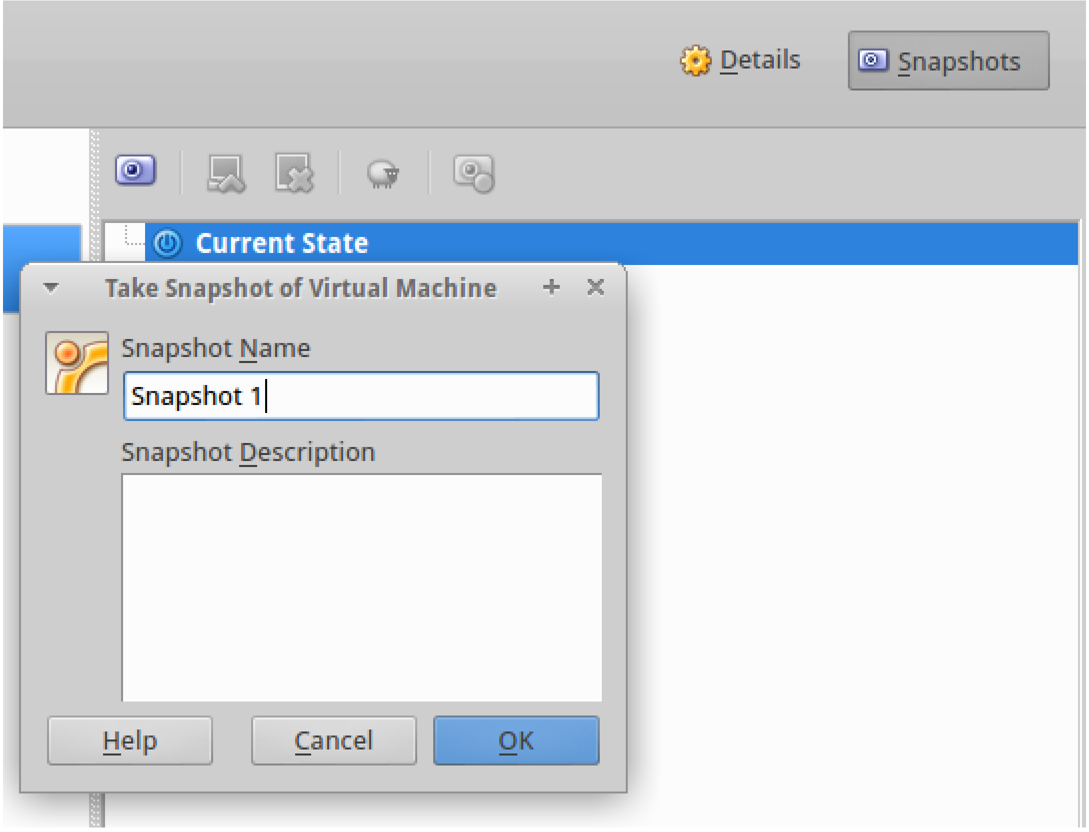
● Full clone

○ Linked clone

Snapshot

We use this feature when trying new applications that have a chance to destroy our system or result in malfunction, when we are not sure if later the VM will work just fine like it was before making the changes. So before making any potentially harmful change, a snapshot is created so that in case of a malfunction the VM can be restored to a previously correctly functioning time. It’s like taking a snapshot of an entire OS files. Like a restore point.
Click on the camera shape icon to begin creating the snapshot:





Snapshot Name: Snapshot 1

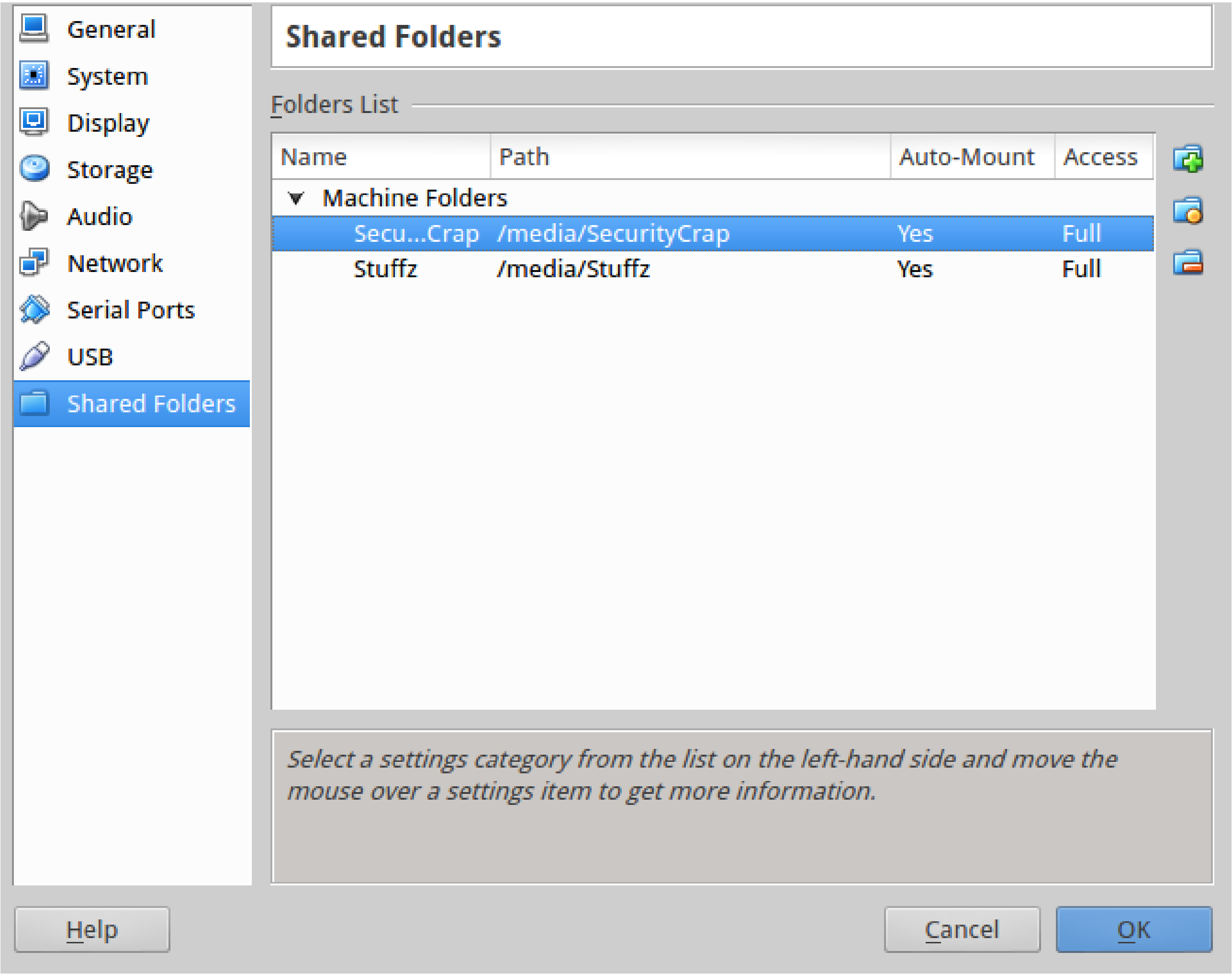
Snapshot Description

File Sharing

Files between your host machine and virtual machine can be shared by using the "Shared Folders" option. On a running VM, click on Devices and select "Shared folders", a window will pop up, on the right of which there will be an add button. It then opens another window asking to locate the folder or a drive that needs to be shared and whether to make it permanent and read-only etc. Choose whatever drive or folder that you want to share from the host-machine. If on a Linux VM, the shared folder will not show up just like that. You will have to issue the following command to mount the shared folder:

```
sudo mount -t vboxsf <Share Name> /mnt
```

The share name can be found from the "Shared Folders" window. The share name should match, otherwise it will show an error that location was not found. Here "/mnt" has been used as the default directory to mount the shared drive, you can use any folder as per choice.
On a windows VM, the shared folder or drive will show up in the My Computer window, under the Network Folders after restarting the VM.



Shared Folders

Folders List

Name	Path	Auto-Mount	Access
▼ Machine Folders			
Secu...Crap	/media/SecurityCrap	Yes	Full
Stuffz	/media/Stuffz	Yes	Full

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

Sharing Devices

Devices like USB, Dongles can be shared in a Virtual Box. On a running VM, click on “Devices”, there will be an option “USB Devices”, it will show all the devices connected to the host machine, including the fingerprint device, pen drives and dongles(data-cards).

Once you select a USB device from there, that device will be unavailable in the host machine and will only be available in the virtual machine.

It’s another way of sharing files using Pen Drives, and a very intelligent way to run various Internet connections at the same time.

Explore the other options in Virtual Box, they are fairly easy to understand. The beauty of it is that it has been designed so that users can enjoy many flavours of OSes without accidentally harming the host machine, so feel free to explore VirtualBox. I am sure you will love it. For any problems encountered, they have their help community. Go to their Website to check their documentation and in case of any problems they have a community forum too.

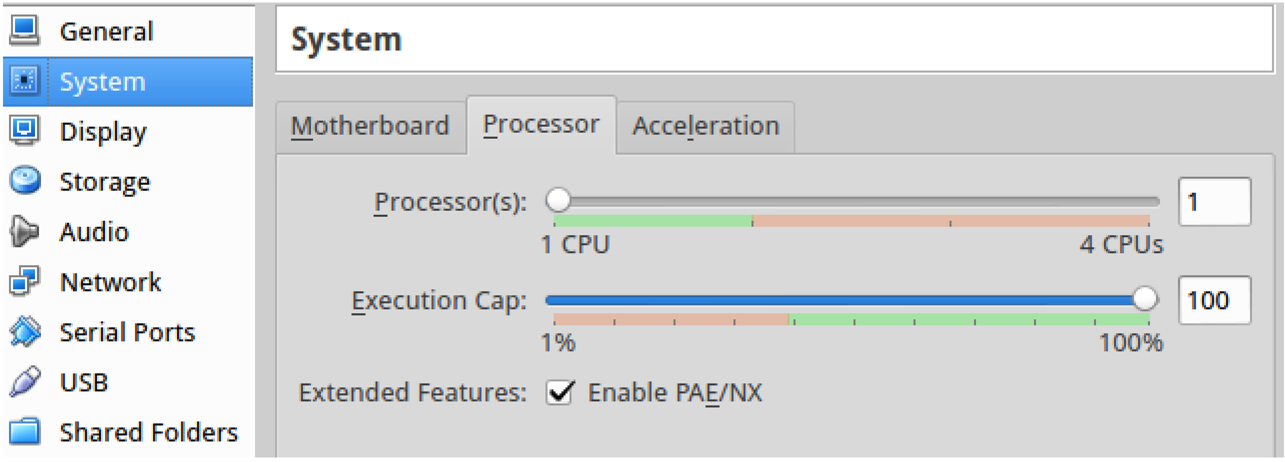
https://www.virtualbox.org

Troubleshooting

While installing Linux as a VM on a windows host machine, users encounter an error saying there is a kernel error or mismatch, something like that.

To get rid of that error, close that VM, click on its “Settings”--> “System” --> “Processor”

You will see a check box in the bottom “Enable PAE/NX”. Check that and restart Your VM, it will run just fine now.



Processor(s): 1

Execution Cap: 100

Extended Features: ☑ Enable PAE/NX

About Ubuntu

Xubuntu = Ubuntu + Xfce

Ubuntu is one of the many Linux distributions, what the distros does is develop their OS on the Linux kernel, which is distributed for free. Then the Linux OS thus created is either distributed for free or at some price. Red Hat is one of the most popular Linux OS that does not comes for free. Open Source does not always means free of Cost. The Source code must always be freely distributed with the product.

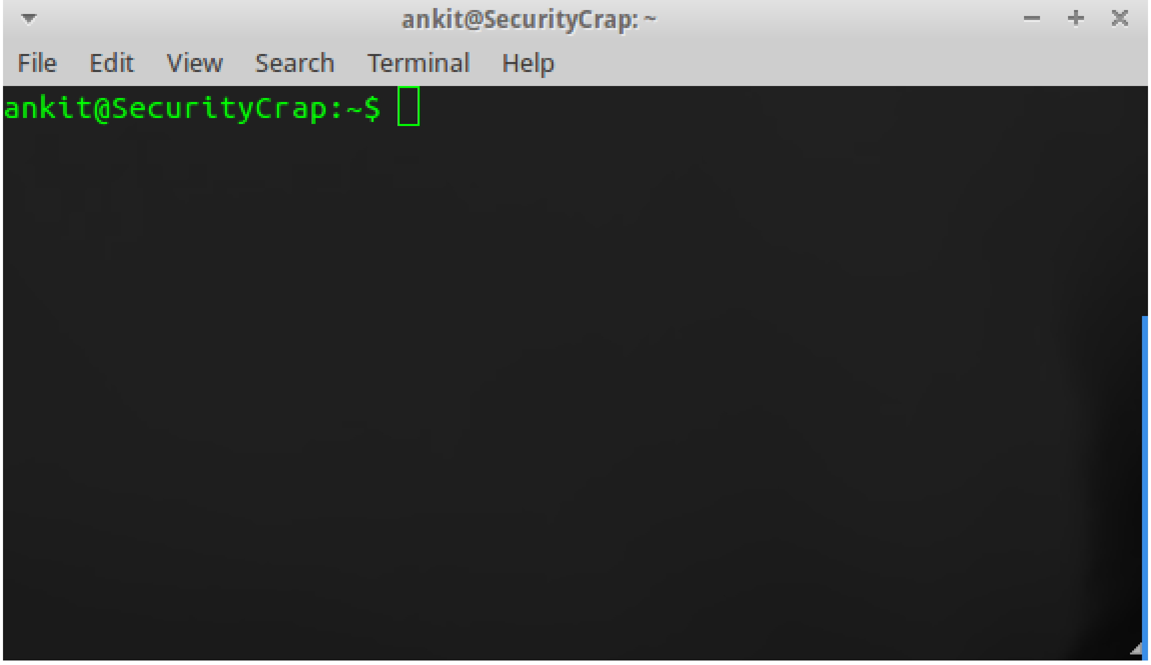
Getting started with Linux

After installation the first thing that you should do is create a root password or disable root.

Root is the administrator account in Linux and it is advised not to log into GUI mode with the root account, also one has to be very careful performing actions when logged in as root. Like, when a non-root user performs some action that requires system privilege, he is prompted with a confirmation of action while root user is not. System just performs actions right away when instructed by root. A simple rm -rf *.* by root could delete all files and you will have to perform a fresh install.

The Terminal

You are asked to create a user during installation. Use the password that you provided to log in. Press alt+F2 and type Terminal, the cli will open. Right click on the Launcher and select 'pin it to Launcher' to have Terminal always present in the Launcher.



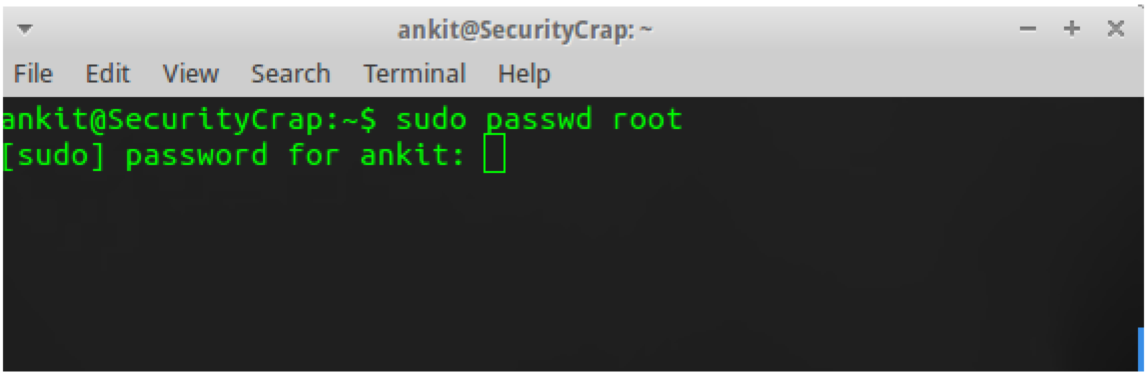
```
ankit@SecurityCrap:~$
```

Now, what you see in the terminal - ankit@Securitycrap:~$

Here ankit is the username that I created during installation, and SecurityCrap is the hostname of the machine. Yeah you can scroll back and see there are different names used in the snapshot, I hope you have used different names too from what was shown. Anyways, moving on..
“$” tells that it’s a normal user(non-root account). When you log in as root the $ sign changes to '#'.

Setting a root password

In the terminal, type - ankit@SecurityCrap:~$ sudo passwd root
You will get the following message -
[sudo] password for ankit:



```
ankit@SecurityCrap:~$ sudo passwd root
[sudo] password for ankit:
```

Type in the password You created, it will stay blank, there won’t be any “*”'s as you type the password, it’s a good security mechanism that does not disclose the length of your password but many times it gets very annoying when you mistype a letter and don't know how many characters you mistyped. So you press the 'backspace' till you are sure it must have erased the wrong/mistyped password entirely and then type it again.
This normal user ankit, which was created during installation gets the privilege of changing root password with sudo rights by default. Other users that you add later won’t have the same sudo privileges, unless that is changed manually, which we will talk about later in the book.

After typing your password, it will ask you to
“Enter new UNIX password:” and then
“Retype new UNIX password:”
Password will be successfully changed only after you get the password right in both.

Now you can switch from normal user to root user in terminal by typing command- su.
It stands for Switch User. If you don't mention the name of the user you want to switch to you will be switched to root user, of course after the correct password.
Once you change to root, the prompt will look like this:
“root@SecurityCrap:~#”

From here you can switch to other users too with
root@SecurityCrap:~# su ankit
Since you are the root user you won’t be asked the password for switching into a normal user.

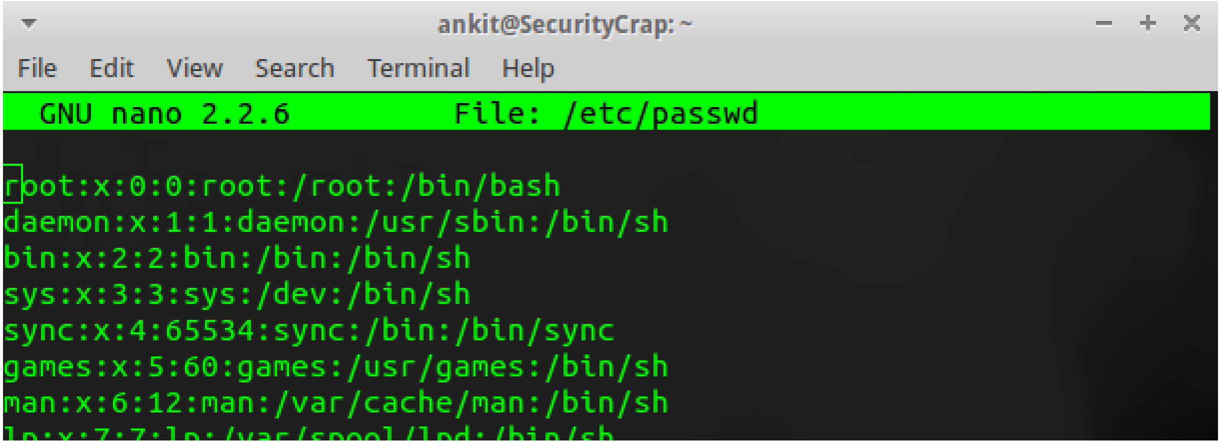
[Tip: The su command by default will switch you to root but will not take you to the root's home directory which is /root, you will stay in the same directory as you are. To switch to root and go to the root's home directory as well, type “su -”.]

[Tip: Other ways to switch to root
sudo -i and sudo -s will also switch you to root user and won’t ask You for the root's password but for Your own non-root user password you are logged in with.]

Disabling root account

Sometimes it is recommended to disable the root account. Since logging in as root can be dangerous sometimes and is not recommended unless you have some experience with Linux, and also to prevent some attack vectors it is advised to disable root. This can be done by -
ankit@SecurityCrap:~$ sudo nano /etc/passwd

This will open the file that contains all the information about all the existing users in the system, the root user will be at the top with user id 0(we will come to that). Change “/bin/bash” on that line to “/sbin/nologin”



```
  GNU nano 2.2.6            File: /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
```

Press 'ctrl + x' and then 'Y' followed by enter to save changes to the file and exit. Now when you try to switch to root it will show an error because now root user will not be getting a shell.

Package Management

Update the system:

```
ankit@SecurityCrap:~$ sudo apt-get update
```

What this does is checks the repository for available updates. No application update will be downloaded, only the information about the available upgrades will be. It will be around 10-20 MB the first time. Post that less than 2 MB.

Now after you have all the information about the software in your system and the available upgrades in the Ubuntu Server, type

```
ankit@SecurityCrap:~$ sudo apt-get upgrade
```

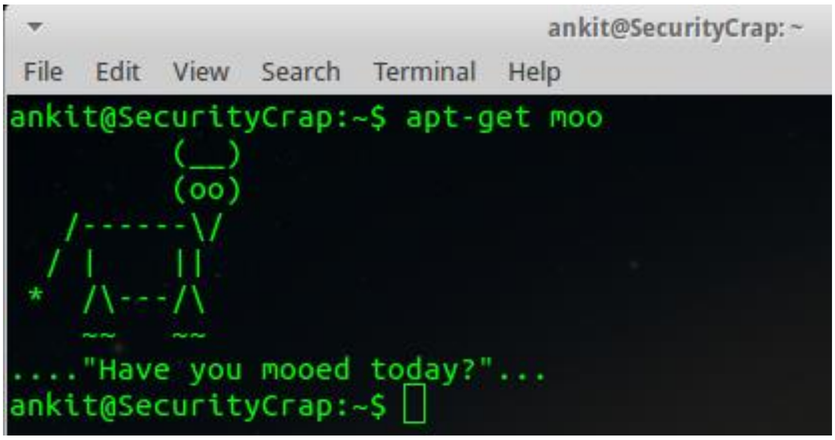
This will prompt you with the names of all applications for which updates are available, the size of the files that it will download and the amount of extra space it will take on your system after installation, sometimes it frees up some space too after installation.

This can also be done graphically, by clicking on the rightmost top settings menu--> Updates Available. It will present a menu consisting of all available updates. To perform apt-get update graphically, click on “Check”, the same thing will happen but graphically. Doing it graphically is boring, and it does not shows how much extra disk space it will consume or free up after downloading and installing updates, at what speed its fetching the updates, but with command line You get to see all of that. Graphically you just sit and wait for indefinite time and miss all the fun.

On any other Ubuntu based OS, if the 'Updates Available' option is not available, look up for 'Update Manager' on the dash or enter update-manager on the terminal and press enter to get the graphical version of the updater.

After it completes the update process, click on “Install” to do the Upgrade part.
There is an Easter Egg in apt-get, try this command:

```
ankit@SecurityCrap:~$ apt-get moo
```



```
ankit@SecurityCrap:~$ apt-get moo
         (__)
         (oo)
   /------\/
  / |    ||
 *  /\---/\
    ~~   ~~
...."Have you mooed today?"...
ankit@SecurityCrap:~$
```

Funny eh? :D
Sometimes when new softwares are installed, old libraries and dependencies become useless and can be removed by the command:

```
ankit@SecurityCrap:~$ sudo apt-get autoremove
```

This will always free up some of your hard disk space.

In case your upgrade was interrupted, sometimes it creates problems when done again. In that case you need to either forcefully install the updates that were downloaded by command

```
ankit@SecurityCrap:~$ sudo apt-get -f install
```

or by reconfiguring the packages with

```
ankit@SecurityCrap:~$ sudo dpkg --configure -a
```

This will solve the problem and you can perform the update and upgrade again.

Many times while updating, You get an error that says “Unable to lock the administration directory (/var/lib/dpkg/), is another process using it? ”
If you read the error message carefully it will be clearly understandable that apt-get is already running. It’s running in the back-end so we can't see it.
You can either kill that process with

```
ankit@SecurityCrap:~$ sudo killall -9 apt
```

or first check the name of the apt running by

```
ankit@SecurityCrap:~$ sudo ps  -A | grep apt
```

and then killing the processes that shows, generally apt and aptd.

Another frequent error is which asks “Are You root?” That means you don’t have the privilege to perform this task. Just add a sudo before the command.

An alternative to apt-get is the aptitude or apt, which is available by default and works pretty much same as apt-get. It only adds a few more additional recommended packages, along with the requested ones.
For update:

```
ankit@SecurityCrap:~$sudo apt-get update can be done with
ankit@SecurityCrap:~$sudo aptitude update
```

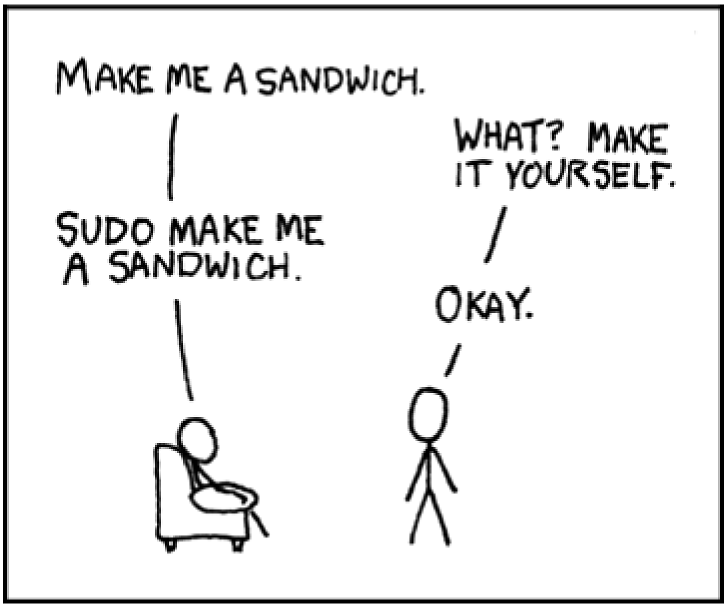
upgrade and dist-upgrade with:

```
ankit@SecurityCrap:~$sudo aptitude safe-upgrade and
ankit@SecurityCrap:~$sudo aptitude full-upgrade
```

Sudo

So what is this **sudo** that does the job of the root?
Well, sudo escalates the privileges of a user and allows to run commands which

otherwise only root could have been able to run.
In other words, it gives an otherwise less privileged user superpowers.
This comic from xkcd will make it even clearer.



There is a file you can edit to control what users can use the sudo command, which is in **/etc/sudoers**.
Find the line that says “# User privilege specification ”

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

As I mentioned earlier in the book that the user we add during the installation by default has this privilege, only the users added afterwards won’t have the sudo privileges.

The sudoer can log into the terminal as root user using his password, even if the root password is not set at all, with

```
ankit@SecurityCrap:~$sudo -i
[sudo] password for ankit:
(enter your password)
root@SecurityCrap:~#
```

or with:

```
ankit@SecurityCrap:~$sudo -s
```

The difference between the 'i' & 's' is that, with the -i option, the home directory is changed to the root user's home directory(/root) while with -s option the user's home directory(/home/ankit) becomes the root's home directory. You can check that using the command “pwd”. Apart from that sudo –s runs the shell specified in the $SHELL environment variable of non-root user as root.

Package Management Continued

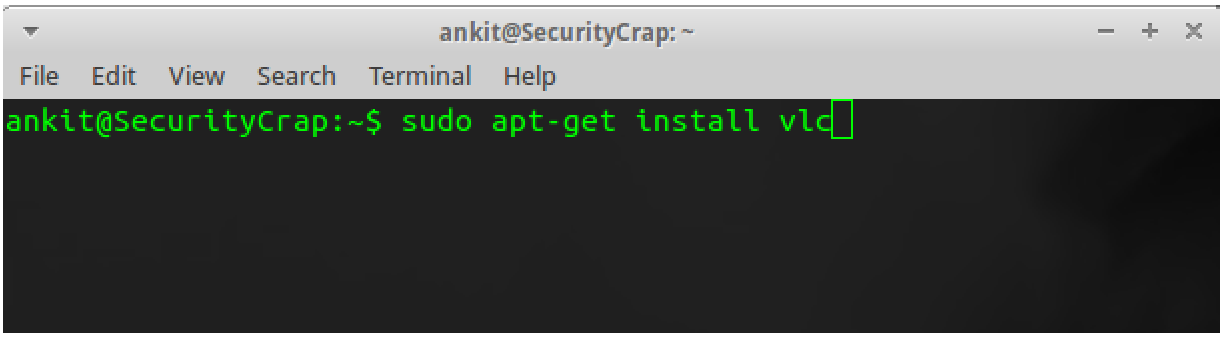
Flash, Codecs & Music/Video Player

What most people want to do next is play music and watch movies, for that Ubuntu has 'totem movie player' and 'Rhythmbox Audio Player'. At first if you try to play music or watch videos without updating the system it won’t support all file types, for that it will automatically prompt you to download the suitable codecs, if You have an internet connection then that will solve the problem. Otherwise, Rhythmbox only plays .flac music files, probably .wav files and totem player plays only .wmv files.

The browser will need flash to play videos online, you can solve that by installing **flashplugin-installer** and the entire codec functionality by installing **ubuntu-restricted-extras.**

Download vlc player as an alternative, which is the most widely used player. It’s also an open source software available for free. Other good options are: banshee, smplayer & kmplayer. Simply fire up the following command to install vlc:

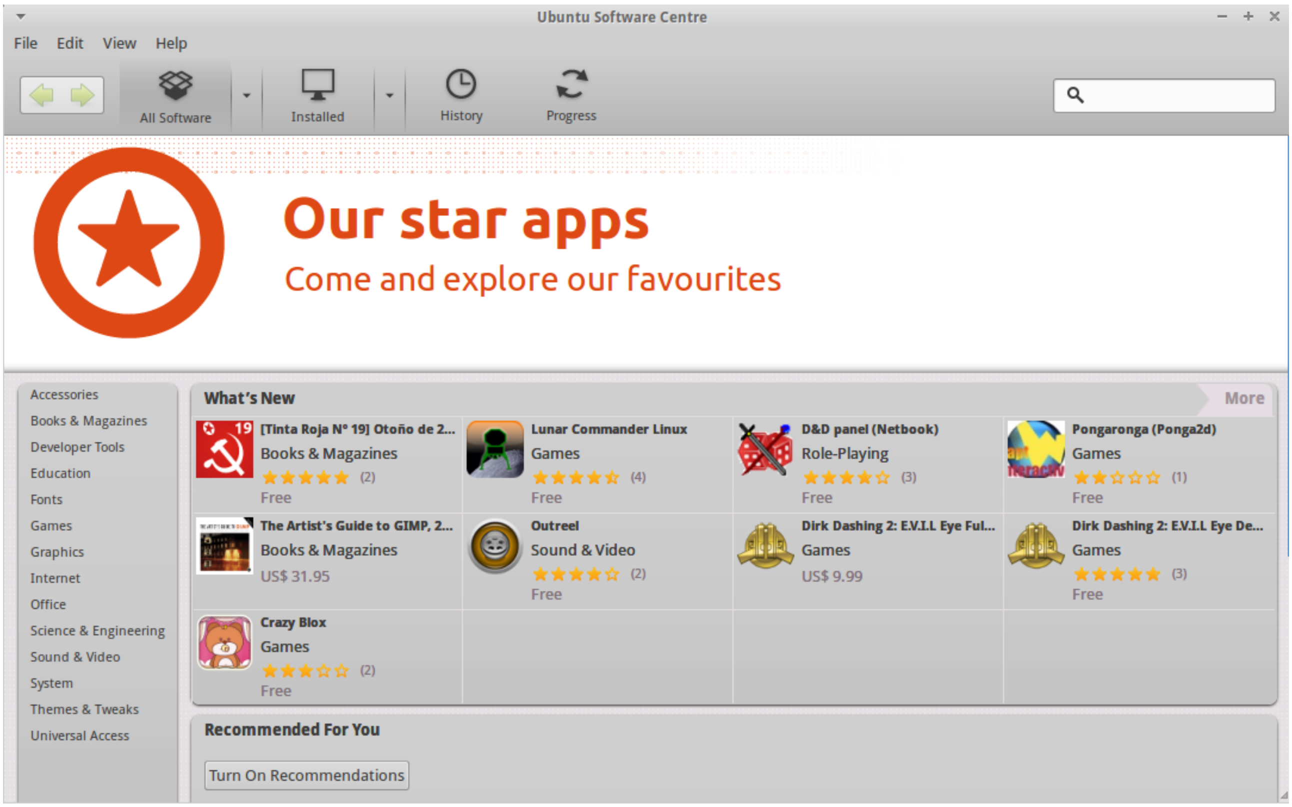
```
ankit@SecurityCrap:~$sudo apt-get install vlc
```



```
ankit@SecurityCrap:~$ sudo apt-get install vlc
```

Installing packages is that easy.

Or you can use the graphical package manager, the 'Ubuntu Software Centre' to do the same:



[Tinta Roja N° 19] Otoño de 2... Books & Magazines ★★★★★ (2) Free

Lunar Commander Linux Games ★★★★★ (4) Free

D&D panel (Netbook) Role-Playing ★★★★☆ (3) Free

Pongaronga (Ponga2d) Games ★★☆☆☆ (1) Free

The Artist's Guide to GIMP, 2... Books & Magazines US$ 31.95

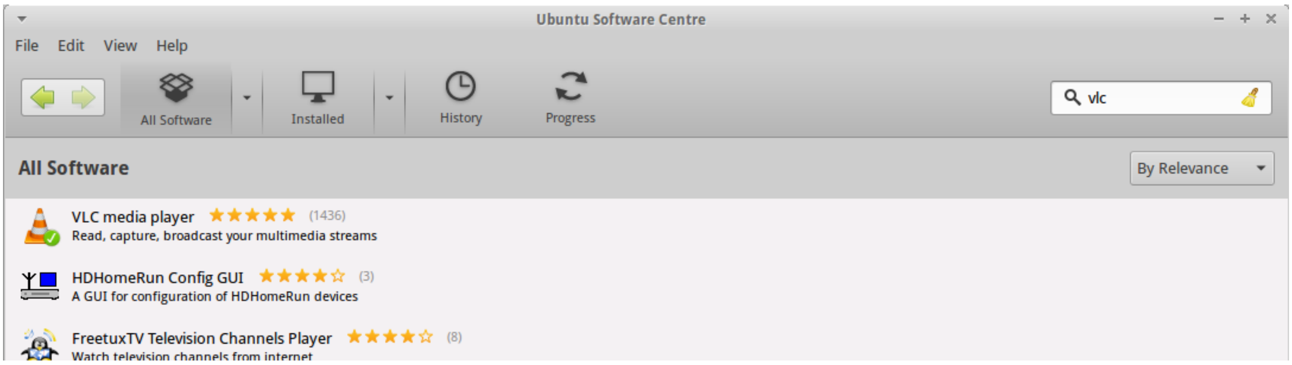
Outreel Sound & Video ★★★★☆ (2) Free

Dirk Dashing 2: E.V.I.L Eye Ful... Games US$ 9.99

Dirk Dashing 2: E.V.I.L Eye De... Games ★★★★★ (3) Free

Crazy Blox Games ★★★☆☆ (2) Free

On the upper right corner, type 'vlc' to search in the repository, and then click on install when the package is found:



VLC media player ★★★★★ (1436)
Read, capture, broadcast your multimedia streams

HDHomeRun Config GUI ★★★★☆ (3)
A GUI for configuration of HDHomeRun devices

FreetuxTV Television Channels Player ★★★★☆ (8)
Watch television channels from internet

You can check the Ubuntu repository if it has that application/software by:

```
ankit@SecurityCrap:~$apt-cache search <package-name>
```

It shows all the matching packages with all the versions available, which could be thousands in number. You can narrow down the result from thousands to maybe hundreds by:

```
ankit@SecurityCrap:~$apt-cache -n search <package-name>
```

Wildcards, such as ^ for beginning and $ for end can be used to further limit the search.

```
ankit@SecurityCrap:~$apt-cache -n search ^<package-name>
```

Once you see the package name in the output copy that and then type the installation command:

```
ankit@SecurityCrap:~$sudo apt-get install <paste the package name here>
```

When the desired application is not available in the Ubuntu repository then the application can be downloaded from external locations and then installed in the system. The application can be available in .deb format which is the debian package format or you can get the entire source code and compile it on your own. Its not as difficult as it sounds.

.deb packages

If You have a .deb package, all You need to do is type:

```
dpkg -i <package-name.deb>
```

and the application will be installed. If it has some dependencies it will prompt you to install those first. If installed with errors, type

```
ankit@SecurityCrap:~$sudo apt-get -f install
```

to remove that package, resolve the dependencies and then try installing again.

.tar ball

Other times you will get the source of the package in a tar file. Tar is used for archiving. Here you first need to extract the tar file and then compile it.
First extraction:

```
ankit@SecurityCrap:~$tar xvf <package-name.tar>
```

then go to the extracted folder with

```
ankit@SecurityCrap:~$cd package-name
```

then find the configure file, and type:

```
ankit@SecurityCrap:~$./configure
```

the "./" is put before any executable, let it be .sh, .py etc given that the libraries to execute already exists. On "ls" the executables show in green color.
After successful configuration type:

```
ankit@SecurityCrap:~$make && make install
```

to complete installation.

Most packages get installed in this manner, if it does not then all such packages comes with a README or/and INSTALL file that contains all the additional instructions one needs to successfully install the application.

.tar.gz and .tar.bz2

These are not much different, just the tar balls compressed. A .gz after .tar suggests the package has been first archived using tar and then the resulting tar file has been compressed with gzip, hence the extension .gz
Here first you need to gunzip it:

```
ankit@SecurityCrap:~$gunzip package-name.tar.gz
```

and then deal with the .tar file.
Or in one step You can issue the following command to both unzip it and untar it:

```
ankit@SecurityCrap:~$tar xvfz package-name.tar.gz
```

The option z tells tar command that the file has been gzipped and so it first gunzips it then extracts the .tar file.
Many people pronounce 'gunzip' as 'gun'-'zip' but actually it should be pronounced as 'jee'-unzip.

The .tar.bz2 file is a tar file than has been compressed using bzip. So first b-unzip it:

```
ankit@SecurityCrap:~$bunzip2 package-name.tar.bz2
```

and then take care of the .tar file or to do it on one shot:

```
ankit@SecurityCrap:~$tar xvfj package-name.tar.bz2
```

PPA: Personal Package Archive

This is the next easiest way to install software and keep it updated. The idea behind creating PPA was to enable developers to get a group of testers, use the beta versions and if they are unhappy with the changes they can perform a clean uninstall too.

*Matt Zimmerman quoted: "**A PPA allows a developer to form a community of testers who are interested in their changes. The testing community can install the packages, run them for the test period and then remove them cleanly from their system.**"*

Ubuntu comes with some default applications like Firefox, document viewer etc. There are tons of other packages available in their repository, those have been tested by Ubuntu community and then added, users can fetch those software by using either the Software Centre or performing an 'apt-get install'. But there are a certain packages that are not supported by them, some not at all while some are only added after a stable release is available and tested by the community.

Let's consider LibreOffice, by default Ubuntu comes with libreoffice but whenever a new release comes, the users are not able to update right then, they will be able to only after it has been tested and put in the repository by the Ubuntu community.
So, to make its distribution easy comes ppa, anybody could simply add a ppa to their list of software lists, which is in the directory '/etc/apt/sources.list.d/', a pgp key is imported, do an apt-get update and then perform installation just like any other package in the main repository.

Xubuntu, doesn't comes with LibreOffice, till date 12.10 was released and there was

no libreoffice there. So, we can either get an old release from the official repository or do a ppa installation for the latest release, another way to get the latest release is by downloading all the .deb files from the official site.
Here is how we do it the easy way with ppa:

```
1. ankit@SecurityCrap:~$sudo apt-add-repository ppa:libreoffice/libreoffice-4-0
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: Total number processed: 1
gpg:            imported: 1  (RSA: 1)

2. ankit@SecurityCrap:~$sudo apt-get update
3. ankit@SecurityCrap:~$sudo apt-get install libreoffice
```

Now the latest version of libre office will be installed.
Same can be done with firefox and other packages too for which the latest release is not made available quickly, just check for their ppa installation and add it as done above.

File System Types

Now since Your Ubuntu is up and running, let's go a bit back where we encountered the different drive formats we came across during installation i.e. ext4 and swap partition.

Like Windows has NTFS which you might be familiar with, Linux supports ext types, ext2, ext3, ext4. Although it does supports NTFS and FAT like windows too but when it comes to installation you need ext type partition or reiserfs etc.

The first filesystem used by Linux was the Minix filesystem-

- had file size limit of 64 MB
- file name limit of 14 characters

Next came the ext(extended) file system-

- solved the previous two problems
- allowed 2 GB of file size
- allowed file name upto 255 characters
- but had modification problems

Then came ext2-

- modified ext
- allowed 2TB of volume size
- but had no journalling system

ext3-

- journalled file system
- adds file system growth and indexing
- take less CPU power

ext4-

- backward compatible to ext3
- extends storage limits
- performance improvement
- supports volume sizes upto 1 Exbibyte(1 million TB)
- file size upto 1 TB
- new block allocation algorithms

Journalling

It basically is helpful during file crashes. It helps to restore the file by keeping track of the changes that are being made in the file but are not committed by the user. In other words, sometimes while working on ppts or docs we forget to save the file as we progress and in case of power failure or something similar we end up losing the changes we made to the file, many time the entire file content too. Journalling can help here. When the system crashes it gets easier to get it back. Files under journalling system are less likely to get corrupted.

FAT(File Allocation Table)-

- used in floppies and USBs
- prominent use of file indexing
- each entry contains the number of next cluster of the file or a marker indicating the end of file.

NTFS(New Technology File System)-

- supersedes FAT
- better performance, Access Control List and Journalling

Basic Commands

We have already discussed some very commonly used commands such as apt-get, tar, bunzip, gunzip etc. Let’s discuss some more:

cd - change directory

During the package installation using dpkg and tar, you need to be either in the directory which contains the packages or mention the exact location. For example if my file is on the Desktop then to extract a tar:

Method 1:

```
cd /home/appy/Desktop
tar xvf nmap6.tar
```

Method 2:

```
tar xvf /home/appy/Desktop/nmap6.tar
```

pwd - present working directory

At any point You can check what directory You are in using this command, this shows the exact location starting from the topmost heirarchy. Example:

```
pwd
/home/appy/
```

For as long as You are in Your home directory the prompt will not show the directory hierarchy, but the moment You change location the prompt will show the location at the end like:

```
ankit@SecurityCrap:/opt/set$
```

This shows that I am in the set directory which is inside the /opt.
The area between the colon “:” and “$” or a “#” shows the present directory you are in.
A tilde(~) sign represents Your home directory.

ls - list

lists the diretory contents. To view the hidden files add '-a'.

man - manual page

It contains the desciption of the command that follows it, includes all the arguments that goes with the command and a small desciption about it, usage and examples.
Use man man to view the manual page of command man itself.
Most commands display the help index by either adding a '-h' or '--help' after the command. Use all these to see the difference.

The unix (man)ual is in the form of man pages which are accessed via the man command. The pages are categorized into numbered sections from 1-8 historically Linux added an extra 9th category for kernel stuff.

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions e.g. /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]

info - information
It takes You the information document of that command. It contains all the minute details of the commands that is not shown in manual page.

whoami - shows Your username
This will display your username. In a bash shell the username is at the beginning of the prompt as shown earlier but in sh shell there is just a # or a $, there whoami can be useful.

mkdir - make directory
This creates an empty directory.

rmdir - remove directory
Used to remove an existing empty directory. To remove a directory that has files in it this command wont work, for that You need to type:

```
rm -rf <directory-name>
```

cp - copy
copies a file/directory from source to destination. To copy directories add '-r' to copy all the contents of the directory and the directory itself recursively.

mv - move
Used to cut and move it to another location.

rm - remove
deletes a file.

There are thousands of commands, keep exploring them and don't forget to see the manual page, information page or the help index of the command before using it.

Fire this command for fun:

```
aplay /bin/bash
```

noisy eh? Use ctrl + c to stop the previous command.
Next try that same command with a & in the end:

```
aplay /bin/bash &
```

Now ctrl+c wont work because with the '&' in the end it tells the command to run in the background. To stop it now You will have to use the kill command.

kill - kill process by id
kill <process-id> kills a process. Process id and process names can be seen by the command ps.

killall - kill process by name
kill requires the process id but killall requires the process name. So to stop the previous aplay command use the following command – killall aplay

There are a few command line editors in Linux:

nano: It is very simple text editor. Just type any name of file with nano to create a new text file and edit it. Write into file and then press ctrl+x to exit, it asks to make changes or leave without making changes, to write press y and then enter, and to just quit without making changes type n and then enter. It has all the options shown at the bottom.

vi/vim: Very powerful editor. It can also be used as a hex-editor. Using vi editor is a bit difficult for new users. Try to get aquainted with it. It runs in two modes, the command mode and the insert mode. Use 'Esc' to switch modes. At first when in a vi editor You are in command mode, type 'i' to enter into the insert mode, now You can write into the editor. When finished, switch to command mode by pressing 'Esc' button and then type – 'wq' then enter to save and quit. Wq will be seen at the bottom of the terminal. To just write and stay type 'w', to quit without saving 'q' and to quit forcefully add an exclamation mark '!' with 'q' i.e. 'q!'
It does not shows options, but there are many options to be used with vi. Explore them.

Shortcuts

Try this:
type ctrl+X followed by ctrl+E in the terminal.
It opens up the default text editor, which is the nano editor in all forms of Ubuntu.

On some other linux systems, it might fire up the vi editor.

Moving Cursor

ctrl+A to get to the start and ctrl+E to get to the end of the line.
ctrl+D or delete button functions the same.
ctrl+K deletes everything from the start to the end.
ctrl+T transposes the last two characters: qrst-> qrts
Esc then T, transposes last 2 words: abcd efgh-> efgh abcd

The default graphical text editor is Gedit in Gnome based distro and kwrite in KDE distros. You can start it from the terminal by:

```
ankit@SecurityCrap:~$gedit &
```

You can start any graphical application from the Terminal in the same way. For starters try firefox & or rhythmbox &.

Another useful command is: **history**
It displays the list of previously entered commands, which are stored in ~/.bash_history file.

```
ankit@SecurityCrap:~$history -c
```

clears us the history but does not clears the bash_history.

Press the up and down key on your keyboard to see the previous and next commands in history,
ctrl+P and ctrl+N does the same respectively.

ctrl+R can be used to search for the previous command, the string supplied need not be the first letter of the command, it will show any command that had those letter supplied.

Tweaking History

Here is a little tip to tweak your history a bit:
You can add timestamp to your history by
adding the line: export HISTTIMEFORMAT="%d/%m/%y %T "
in ~/.bashrc or /etc/profile whichever seems to work.
Everytime the bash is started, it reads from the .bashrc and the /etc/profile file.

The Linux File System

The Linux filesystem is derived from the Unix filesystem. You can see all the entire hierarchy of the Linux file system by issuing the following command: “man hier”

The forward slash - / is the main directory in which all the other directories exist. This is called the root directory. Inside the root lies the following directories that we are going to discuss.

/bin - the binaries directory. It contains all the executable binaries for the commands that are executed by both the unprivileged and privileged user. It has the files that can be used to repair the system.

/boot - this contains the files that are needed during the boot process.

/dev - All the physical devices - the internal hard disk drive, the USB drives and dongles that are plugged into can be found here. The devices directory and hence the name.

/etc - configuration files. All the software, daemons, applications configuration files are stored here.

/home - home folder of all the users in the system.

/lib - all shared libraries needed for an application or a command to run gets stored here.

/media - drives are mounted temporarily or permanently here, including the removable drives.

/mnt - used to mount files temporarily.

/opt - optional files also called addons are stored here.

/proc - files that provide information about the running processes and the kernel are stored.

/root - home directory of the root user.

/sbin - system binaries are stored here that are used during the system boot, normal users cannot execute the commands from here.

/tmp - temporary files which are deleted every now and then.

/usr - it contains a large number of directories that contains manual pages, local programs, c and c++ libraries, executables that are not required during boot process and that can be executed by both normal and root user, and many other files.

/var - contains the files that vary in size, has the mail folder.

For further detailed explanation type the command mentioned earlier, which was:
# man hier.

File Permissions

Now that the file system has been taken care of let's move on to a very interesting topic of file permissions.
When a file is created it is created with the permissions of the user who created it. It is assigned the name of the user who created it and the name of the group of users that can use it with time stamp, file size etc. Many times there are files which a normal user is unable to open, read or write just because he does not have the permission. This happens with the less privileged accounts. The root user can read, write and execute all files, there is no stopping the root user. That is why many OSes doesn't allow root login by default.

Whenever you get errors like the ones mentioned above, unable to copy and paste some file in the GUI, consider taking a look at the permissions of the files by issuing the following command:

```
# ls -l
```

or

```
# ll
```

the -l and the command ll is for long listing, You will see something like this:

```
.....
-rw-rw-r--.         1 ankit ankit        362282427 Jun 26 17:04 SIA302.mp4
drwxrwxr-x.         2 appy appy          4096 Jun 18          23:42 Ubuntu One
drwxr-xr-x.   4 appy appy          4096 Sep 17          11:55 Videos
drwxrwxr-x.         12 appy appy         4096 Sep 27          17:37 VirtualBox VMs
drwxrwxrwx.         4 root  root         4096 Sep 13          00:15 workspace
.....
```

The "d" denotes that it is a directory or a folder. Files like text files or movie files etc will not have a "d" in the beginning.

The next 9 characters denote the r=read, w=write and x=execute permissions. It's a pattern of 3 characters- rwx.
The first block of rwx is for the owner of the file. The next block is for the group and the last rwx is for all the other users in the system.
If read permission is allowed there will be a r, if not there will be a dash "-".
Similar with the write and execute.

Changes in the permission of a file can be done in the following two ways using the chmod command:

- Alpha   u+x, ugo+rwx, a+x

- Octal 644, 755, 777

In Alpha method You mention the categories first which are to be allowed which are: u=user(self), g=group, o=other users and a=all users, followed by the permissions which are like mentioned before r=read, w=write and x=execute.

In Octal method we use numbers, each permission is assigned a number just like the binary values, the first from right which is x=1, then w=2 and r=4.
So to assign a rwx to the user(self) we will sum up r, w and x which is 1+2+4=7, and similarly for the groups and other users.

7 = 4(read) + 2(write) + 1(execute)
6 = 4(read) + 2(write) + 0(execute)
5 = 4(read) + 0(write) + 1(execute)
4 = 4(read) + 0(write) + 0 (execute)
3 = 0(read) + 2(write) + 1(execute)
2 = 0(read) + 2(write) + 0(execute)
1 = 0(read) + 0(write) + 1(execute)
0 = 0(read) + 0(write) +0(execute)

I just wanted to do that, make sure you fully understand that math in file permissions.
So for 754, the first number from the right(4) is for the other users which are not the owner nor in the group, second(5) is for the user's group mentioned and the third one(7) is for the owner himself.

Example:

```
-rw-r--r--. 1 appy appy 396 Aug 13 22:03 honeyd.sh
```

With the command: # chmod 755 honeyd.sh
Output for
# ll honeyd.sh is

```
-rwxr-xr-x. 1 appy appy 396 Aug 13 22:03 honeyd.sh
```

It being a file there is no d at the beginning. The first 7 in the previous command resulted in the first rwx which is for the user appy, the second 5 resulted in r-x, read and execute permission for the users in the group appy and the last 5 resulted in r-x, read and execute permission for all the other users.

Change group of the file

```
ankit@SecurityCrap:~$chgrp <group_name> filename
```

Change owner of the file

```
ankit@SecurityCrap:~$chown new_owner_name:group_name filename
```

Practice it a bit.

The output we see in ll command or ls -l, there can be the following in the first character:

- b denotes a block special file-that move data in blocks
- c denotes a character special file- through which system transfers data
- l denotes a symbolic link
- p denotes a named pipe – during IPC.
- s denotes a domain socket – IPC(Inter Process Communication) on same host.

Moving ahead to the second field that is a number. The number denotes the total number files in it, it will be 1 in case of a file and more than 1 in case of directories.

The next field in:

```
-rwxr-xr-x. 1 appy appy 396 Aug 13 22:03 honeyd.sh
```

the first appy is the user who created the file, the owner and the second appy is the name of the group that it belongs to. When the file is created it is assigned the same names in the owner and group field, although it can be changed, we will discuss that soon.

The next field which is 396 here which is the byte count of the file. In other words the size of the file in bytes.

Other fields are self-explanatory, but still let's cover that too.

Aug is the month, then there is the date field which is 13 here, then the time in hours:minutes format and finally the name of the file.

In file permissions, there are three more permissions that are worth discussing here:

- setuid - file(or command) with SUID set enables users to be treated temporarily as privileged when run
- setgid - files under a directory with SGID will inherit the directories properties
- sticky bits - prevents users from renaming, moving or deleting contained files owned by other users.

Example:

- passwd command has SUID set, which allows changing password in /etc/shadow for anyone which only root can.

```
chmod u+s "/<command path>/command"
```

- sgid on commands run with group id of the owner

chmod g+s "/<path>/comand"

- Sticky bits enabled files can only be deleted by root or the owner

chmod o+t /<path>/directory

Two more options of interest here are: **--reference and -R**.
With chmod --reference <filename1> <filename2> you can assign permissions to filename2 same as filename1, saves you time when dealing with multiple files at a time.
And -R is for recursively assigning a permission to each and every file in a directory and the sub-directories lying under it. E.g.-
ankit@SecurityCrap:~$chmod -R 600 /home/ankit
to change every file under the home folder readable/writable by owner.

Some additional commands

du <filename> - shows the disk space occupied by in kilobytes
du without a filename will go on listing every single file and directory and all files inside the directory.

df - shows free space

free -m - shows free memory

date - shows the current date

finger - shows all the users currently logged on into the system

head <filename> - prints the beginning of a file

tail <filename> - prints the end of a file

locate <filename> - searches for the filename and displays its location

grep <pattern> - looks for the matching pattern in the entire system, it is not advised to use grep like this, it can go on for hours because it goes inside every file looking for the matching pattern and returns it with the filename.

Use grep in combination of some other command using pipelining.

Pipelining

Pipelining is when you connect two commands, where first a command gets executed and then the next command is executed on the output of the first command. You can make bigger chains further pipelining it.
Example:

```
# ls -l | grep honeyd.sh
```

The symbol after ls -al is the pipe symbol. What it will do is first list all files in the current directory and then out of all the files in the output it will look for the pattern honey.sh and then display it.

Input and output redirection

In output redirection, you send the output into a file and don't see the output in the screen with > sign.
Example: ls > list.txt
This will create a file named list with txt format and store the listing of the directory. No result will be displayed.

1>filename
Redirects standard output to the file "filename."
1>>filename
Redirects and appends standard output to file "filename."
2>filename
Redirects standard error to file "filename."
2>>filename
Redirects and appends standard error to file "filename."
&>filename
Redirects both standard output and standard error to file "filename."

Here:
1 is the file descriptor for standard output
2 is the file descriptor for standard error output
0 is another important file descriptor for standard input.
These 3 file descriptors are always open when a program runs.
In *nix, everything is a file and each file has a file descriptor associated with it.

Now, the input redirection uses the sign less than "<".
It takes input from a file.
Example:

```
# grep appy < appy_biography.txt
```

Will look for the word appy in the filename appy_biography.txt and show the entire line that contains it.

The cut command is another important command, if you ever need to keep tons of network log files this command mixed with grep is a life-saver.

Cut - Used to remove columns from a file and present the required field.

Let there be a file sample.txt with content:

```
Column1 Column2 Column3 Column 4.
This is line one.
This is line two.
This is line three.
Demonstrating the cut command.
```

```
cut -d ' ' -f1sample.txt
```

will divide the text into columns, taking the spaces as delimiter, cut off everything from the text and just present you with the first column in each line in the text. Hence, the output:

```
Column1
This
This
This
Demonstrating
```

```
cut -d ' ' -f2 sample.txt
```

```
Column2
is
is
is
the
```

Now lets set '.' as delimiter, which is present in the end of each line.

```
cut -d '.' -f1 sample.txt
```

```
This is line one
This is line two
This is line three
Demonstrating the cut command
```

because everything before the '.' comes as field 1, while there is no field 2, so if we supply any other field number here, we will be presented with blank spaces equal to the number of lines in the sample.txt text.

Without setting the delimiter, cut can output the desired characters from each line given in a text.
cut sample.txt -c1-5

```
Colum
This
This
This
Demon
```

Managing Partitions(Devices)

Most of the new linux users are the ones who have just migrated from windows and managing their pre-existing partitions is a bit problematic. It's because the little bit complicated file system in Linux. Simple because there is no "My Computer" like in Windows, that shows all the other partitions.
So here is what you do.

Open Terminal, switch to root with the command su. Then
# fdisk -l
To get this:

```
 Device     Boot  Start      End        Blocks     Id  System
/dev/sda1   *     2048       58630143   29314048   83  Linux
/dev/sda2         160196606  625141759  232472577  5  Extended
/dev/sda3         58632192   160194559  50781184   83  Linux
/dev/sda5         160196608  164159487  1981440    82  Linux swap / Solaris
/dev/sda6         164161536  359530495  97684480   83  Linux
/dev/sda7         359532544  625141759  132804608  83  Linux
```

You may get different device numbers with different tags in the System columns, mostly NTFS/HPFS.

Now understanding each column in the output:
The /dev/sda1, sda2 represent the partitions of the hard disk space attached to the computer. The primary storage device which is Your internal HDD takes the 'a' in 'sda'. Insert a secondary storage device, it will be shown as 'sdb1' in '/dev', sdb1 and sdb2 if it has two partitions.
The '*' in the boot column is the boot flag. The system will be booted up with that device. We will come to the how to do it too shortly.

The next column – Start and End represent the starting sector number and end sector number. Each sector is of 512 bytes. You can calculate the exact space in Gigs by doing the math. Subtract the End sector number with the starting sector number, multiply that with 512, thats the storage space in bytes.
Coming to the Blocks column, the number in this column represents the storage capacity in KiloBytes, do the math.

Now, after you know the device names of all your existing partitions, create directories in ' /media '. For example, if You want to mount /dev/sda7 partition permanently as 'Stuffs', create a folder named Stuffs in '/media':
#mkdir /media/Stuffs
Create folders in media accordingly and then open the fstab file to mention what drives needs to be mounted during the boot process.
#nano /etc/fstab
You get this:

```
 /etc/fstab: static file system information.
#
# Use 'blkid -o value -s UUID' to print the universally unique identifier
# for a device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    nodev,noexec,nosuid 0       0
```

The last commented line lists the format that You have to follow in order to permanently mount your drives.
Taking our example, we type there :

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    nodev,noexec,nosuid 0       0
```

```
/dev/sda7    /media/Stuffs        ntfs-3g         defaults        0        0
```

ntfs-3g in type column represents a NTFS partition which is a typical Windows partition. Linux partitions are generally 'ext3' and 'ext4'. Do that accordingly.

The options column is very important but the defaults option just takes care of the importance and the complications too. Options like ro(read-only), rw(read-write), exec(execute permission on the binaries), noexec, user(any user can mount) and nouser(only root can mount) & a few more options.

The dump option checks if the file system needs to be backed up or not, 0 tells it not to back up.
The pass option is a disk check option, 'fsck'. If set to 1, it will always check that device for errors while booting up the system.

Since fsck came up:
Q: “What did the Linux partition tell to the Windows partition?”
A: “Go 'FSCK' Yourself”
=D

After adding all the existing partitions in the fstab file, type

```
#mount -a
```

This will mount all the devices listed the the fstab file to the corresponding folder with the given options.
Now You don't need to worry about the random lengthy partition names that used to get mounted temporarily.

Little more on devices
The devices under /dev have three attributes:
1. Device type   2. Major Number   3. Minor Number

The Major number determines the driver to access and the minor number allows drives to differentiate between similar physical devices.
For e.g.

```
        # ls -al /dev
crw-------.  1 root root       5,   1
```

'c' here represents a character device, 5 is the major number and 1 is the minor number.

/dev/hda represents the primary IDE type disk, and the present day disk which is SCSI, SATA or USB is represented by /dev/sda.

Two more important entries worth mentioning are: /dev/tty and /dev/pts.
The virtual consoles provide text logins through kernel's VGA drivers and are identified as /dev/tty while pseudo-terminals are used by graphical-terminals and ssh-server created by a special VFS, devpts, under /dev/pts.

Connecting to the Internet using a data-card:

Every device gets listed in the /dev. Using data-cards is not so user friendly in Linux simply because most of those devices are not meant to be used in Linux, just like the NTLDR loader is not supposed to support linux. It's one of the biggest reasons given by Windows users for not moving to Linux. But there is a solution, there always is and that too a free solution.

Many distros, like Ubuntu, comes with the default settings to easily connect to the data-cards, it only depends on the device if it's recognized automatically or not. We will discuss a bit about those which take some extra effort to connect. Those come with a CDROM that needs to be ejected, some comes with a .deb package that does not works on 64-bit distros at all.
Check /dev for ttyUSB0, ttyUSB1, ttyUSB2 or ttyACM0, ttyACM1, ttyACM2. If any of these two are listed automatically after inserting the data card then its good. Otherwise, try ejecting the dvdrom from the disk utility, maybe then you can find one of the tty entries mentioned in the /dev.
There is a very powerful script 'wvdial' that makes connecting to the internet very simple. After you get a ttyUSB0 or a ttyACM0 in /dev fire up the following commands:

```
# wvdialconf
# wvdial
```

This will configure the settings and will establish a connection and the output will be:

```
 Carrier detected.  Starting PPP immediately.
```

Followed by IP address, DNS address etc.

If it fails to connect, then it will prompt you the missing parameters that needs to be filled in the /etc/wvdial.conf file.
The configuration for every single carrier is listed on the web.
We will take aircel carrier as an example.
Create a profile for aircel in /etc/wvdial.conf

```
#nano /etc/wvdial.conf
```

```
[Dialer aircel]
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Init3 = AT+CGDCONT=1,"IP","aircelgprs.pr"
Stupid Mode = 1
Modem Type = USB Modem
ISDN = 0
New PPPD = yes
Phone = *99#
Modem = /dev/ttyUSB2
Username = Aircel
Password = aircel
Baud = 460800
```

save & exit.

Now run the command:

```
# wvdial aircel &
```

The bottom lines in the output will be:

```
local  IP address
--> pppd: 8#X  "X `!X
--> remote IP address
--> pppd: 8#X  "X `!X
--> primary   DNS address 8.8.8.8
--> pppd: 8#X  "X `!X
--> secondary DNS address XX.XXX.XX.XXX
--> pppd: 8#X  "X `!X
```

Enjoy surfing!

Not all data-cards support the fancy application in Linux like the ones you get to see in Windows. Some do while some are a huge pain in the butt.
Wireless and LAN connections work just fine.

Devices like pen drives and other removable media might not get mounted automatically in the /media, although on my systems they do. You can find out about its listing in the '/dev' with the help of the command:

```
#fdisk -l
```

Since that is a secondary storage device, it will take up 'b' in the previously discussed

'sda' or 'hda'. Then you can manually mount it anywhere you want.
Let’s say, we insert a pen drive and “fdisk -l” shows it as
/dev/sdb1
To view its contents we can mount it in /mnt by:

```
# mount /dev/sdb1 /mnt
```

Transfer files and when done, unmount it by:

```
#umount /mnt
```

And that's how you safely remove removable drives.

Running exe files with Wine

I sincerely hope none of you are missing windows. If you are, well then get used to it. I can't assure that you can run all those windows programs in linux too, although there are a few useful programs that can be run using this tool called wine. It creates a virtual environment for the exe files to run on linux platforms. We have tried the very popular game named Counter Strike on linux using wine and it ran pretty well, also MS Office 2007, but please get acquainted to OpenOffice or LibreOffice. MS office in linux doesn't feels the same as in Windows. Try running small programs with wine.
First, install this piece of software by running the following command:

```
sudo apt-get install winetricks
```

Now start with small files like notepad.exe, notepad++.exe and calc.exe

Please don't tell me you miss Internet Explorer in Linux.

Using the command line, just add wine before these apps(other than IE please):

```
wine calc.exe
```

or right click the app and open it using wine.
It’s that easy.

Archiving, Compressing & Encrypting Files

The program 'tar' in linux is a very useful for creating and transmitting an archive file. An entire project work can be archived into one file to make it easy for transfer and also to save the disk quotas.

Let's start with some basic examples:

Create a directory and then a few files.

```
ankit@SecurityCrap:~$ mkdir archive
ankit@SecurityCrap:~$ cd archive/
ankit@SecurityCrap:~/archive$ touch a b c d 1 2 3 4
ankit@SecurityCrap:~/archive$ mkdir folder
ankit@SecurityCrap:~/archive$ cd ..
```

Tar

Now creating a tar ball:

```
ankit@SecurityCrap:~$ tar cvf archive.tar archive/
```

```
                #where:     -c = create a new archive
                            -v = verbose mode
                            -f = name of the file
```

archive.tar will be created.

To extract it, we just need to replace 'c' by 'x':

```
ankit@SecurityCrap:~$ tar xvf archive.tar
```

Navigate to the folder just untar'ed and there are all the files.

Zip

'zip' is another program to archive and compress files. Usage is pretty simple:

```
ankit@SecurityCrap:~$ zip zipfile archive
```

```
                #where      zipfile is the name of the new zip file created.
```

and to unzip:

```
ankit@SecurityCrap:~$ unzip zipfile.zip
```

Upon unzipping, the folder will acquire the original name the zip was created from i.e. archive

'tar' supports compression too, gzip and bzip2 type compression.

Gzip

```
ankit@SecurityCrap:~$ tar cvzf archive.tgz archive/
```

and then gunzip and extraction at the same time to get the files back:

```
ankit@SecurityCrap:~$ tar xvfz archive.tgz
```

Another gzip format is the .gz identical to the previous one:

```
ankit@SecurityCrap:~$ tar cvfz archive.tar.gz archive
```

extraction and unzip can be done with the same command used in .tgz:

```
ankit@SecurityCrap:~$ tar xvfz archive.tar.gz
```

bzip2

It uses compression using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding. The compression is a bit more powerful than the others discussed earlier.

```
ankit@SecurityCrap:~$ tar cvfj archive.tar.bz2 archive
```

then extraction and unzipping with:

```
ankit@SecurityCrap:~$ tar xvfj archive.tar.bz2
```

Password Protecting files

ccrypt

Using ccrypt and gpg.

```
ankit@SecurityCrap:~$ ccrypt archive.tar.gz
Enter encryption key:
Enter encryption key: (repeat)
```

and to decrypt:

```
ankit@SecurityCrap:~$ ccrypt -d archive.tar.gz.cpt
```

gpg

gpg is an openpgp encryption and signing tool. It’s based on asymmetric key encryption where you need the public key of the receiver the file is supposed to be sent to, because only the receiver shall be able to decrypt the file with his private key which nobody else has. Or, encrypt it with your own public key, so that it can be decrypted by the private key that you own.

```
ankit@SecurityCrap:~$ gpg -e rarchive.tar.bz2
You did not specify a user ID. (you may use "-r")
Current recipients:
Enter the user ID.  End with an empty line:
```

Enter the id, your own or the receiver's.

Gpg is an amazing encryption for email exchange, not only to encrypt the message but the attachments too.

Password Protecting Zip Files

zip supports a built-in encryption.

```
ankit@SecurityCrap:~$ zip -e archivepassprotected.zip archive/
Enter password:
Verify password:
```

and to unzip:

```
ankit@SecurityCrap:~$ unzip -P password archivepassprotected.zip
Archive:  archivepassprotected.zip
   creating: archive/
```

The Shell Almighty

Shell is an interface between the user and the Unix/Linux system, through which we enter commands that is executed by the Operating System. It can be called as a High Level Programming Language for Unix/Linux.

Whether you are just a beginner trying to install some heavy software resolving every dependencies or a system admin who has millions of Megabytes of data to analyse, knowing a bit of bash-script-fu can come in handy.

Let's start off with a bit of history on the different types of shells.

sh	-	The original shell
csh	-	C shell
tcsh	-	Extended version of C shell, adds command completion, a command line editor
ksh	-	Written by David Korn, far superior functionality than Bourne Shell
zsh	-	like ksh, csh, and bash thrown together
bash	-	The most popular, and the most user-friendly, the Bourne Again Shell which we will be working on.

As we already are aware of the input/output redirections and pipelining, let's quickly move on to creating our first script.
Fire up nano and let's write a small script to print Hello World.

Hello World

```
ankit@SecurityCrap:~$nano HelloWorld
```

```
#!/bin/bash
echo “Hello World”
exit 0
```

Then, ctrl+x and y enter to save and exit.
Next, we will grant this script executable permissions with:

```
ankit@SecurityCrap:~$chmod +x HelloWorld
```

```
ankit@SecurityCrap:~$./HelloWorld
Hello World
```

And, there we have our first script.

Visting the code:

the '#!' in the beginning tells the bash shell that the next argument is the absolute path to the program that will execute whatever follows. Generally, the argument works well with 32 characters. So when you have the program somewhere else and the script fails to work correctly then its probably the exceeding character limit. But on most linux machines it works fine with more than 32 characters.

Echo is as you guessed, prints what we typed next inside the double quotes.

The exit 0 at last is not necessary but its considered to be a good programming practice to have your script exit appropriately with proper exit codes.

Another way to make our script executable is by adding an appropriate extension, .sh in our case. With a .sh in the end, we can skip the chmod part and execute it with the command sh.

Create another file HelloWorld.sh or just copy the old script to this new name script.

```
ankit@SecurityCrap:~$sh HelloWorld.sh
Hello World
```

The benefit of making the script executable, let it be python or bash script, when the program argument (#!) is mentioned appropriately, the bash shell will know which program exactly to use to execute that particular script.

And when you are certain what program the script is written in, you can always fire up that program and pass the script file as an argument.

Executing bash-scripts can be as simple as typing into the terminal and executing interactively. The bash shell recognizes the syntax and the prompt changes from $ to >.

```
$ for i in one two three; do
> echo $i
> done
one
two
three
```

Declaring Variables

Declaration of variables before using them is not required in bash-scripts. It is created when we use them. By default all variables are stored as strings.

```
ankit@SecurityCrap$ name=ankit
ankit@SecurityCrap$ echo $name
ankit

ankit@SecurityCrap$name=1
ankit@SecurityCrap$echo $name
1
```

Using Quotes

Using double, single quotes and backslash as escape character.

```
ankit@SecurityCrap:~$ variable="My var"

ankit@SecurityCrap:~$ echo $variable
My var

ankit@SecurityCrap:~$ echo "$variable"
My var

ankit@SecurityCrap:~$ echo '$variable'
$variable

ankit@SecurityCrap:~$ echo \$variable
$variable
```

Environment Variables

Everytime a shell script runs, some variables are initialized. These are mere informations that are assigned from the values in the environment such as:

```
$HOME     -     Home directory of the current user
$PATH     -     List of directories to search for commands
$0        -     Name of the shell script
$#        -     Number of parameters passed
$$        -     Process ID of the shell script
```

```
ankit@SecurityCrap:~$nano HelloWorld.sh
#!/bin/bash
echo "Hello World"
echo "The Home directory of the current user is:"
echo $HOME
echo "The name of the shell script is:"
echo $0
echo "Number of parameters passed:"
echo $#
echo "Process ID of this script is:"
echo $$
exit 0
```

```
ankit@SecurityCrap:~$sh Helloworld.sh
Hello World
The Home directory of the current user is:
/home/ankit
The name of the shell script is:
Programming/HelloWorld.sh
Number of parameters passed:
0
Process ID of this script is:
26487
```

Parameter Variables

When some parameters are provided with the script, some additional variables are created.

$1 - The first variable passed
$2 - The second variable passed
....and so on
$@ - List of all parameters in a single variable

```
ankit@SecurityCrap:~$nano Helloworld.sh
#!/bin/bash
echo "Hello World"
        set 1 2 3
echo "All the parameters passed are:"
echo $@
echo "First parameter passed is:"
echo $1
echo "Third parameter passed is:"
echo $3
echo "Second parameter passed is:"
echo $2
exit 0
```

```
ankit@SecurityCrap:~$sh HelloWorld.sh
Hello World
All the parameters passed are:
1 2 3
First parameter passed is:
1
Third parameter passed is:
3
Second parameter passed is:
2
```

Conditional Statements

The syntax is pretty simple, a condition check is performed before a certain action is performed.

```
if <condition>
then
        <set of actions of perform>
fi
```

```
ankit@SecurityCrap:~$nano ifcondition.sh
#!/bin/bash
if [ -f HelloWorld.sh ]; then
      echo "HelloWorld.sh found... Executing"
      sh HelloWorld.sh
fi
```

```
ankit@SecurityCrap:~$sh ifcondition.sh
HelloWorld.sh found... Executing
Hello World
```

Make sure there are spaces as in the program, just before and after the opening and closing square brackets. Often the script returns an error when you get that wrong.

The “-f ” we used here is one of the many file conditionals. A few more are listed below:

-e filename	-	true if the file exists
-f filename	-	true if the file is a regular file
-d filename	-	true if the file is a directory

```
-s filename  -      true if the file has non-zero size
-r filename  -      true if the file is readable
-w filename -       true if the file is writable
-x filename  -      true if the file is executable
```

Control Structures

If

```
ankit@SecurityCrap:~$nano ifcondition.sh
#!/bin/bash
echo "Enter the name of the file to search for:"
read file
if [ -f $file ]; then
	echo "$file found... Executing!!"
	sh $file
elif [ -f $file.sh ];then
	echo "$file.sh exists... Executing!! "
	    sh $file.sh
else
echo "File not found.. Exiting!!"
fi
```

In the particular directory where we will be executing the script already has a small script called HelloWorld.sh.

```
ankit@SecurityCrap:~$sh ifcondition.sh
Enter the name of the file to search for:
HelloWorld
HelloWorld.sh exists... Executing!!
Hello World
```

For

The for construct is used to loop through a set of values.
Syntax:

```
for <variable> in <values>
do
	<statements>
done
```

```
ankit@SecurityCrap:~$ nano forexample.sh
```

```
#!/bin/bash
for i in Ankit Prateek Founder SecurityCrap
do
	echo $i
done
exit 0
```

```
ankit@SecurityCrap:~$sh forexample.sh
Ankit
Prateek
Founder
SecurityCrap
```

While

Syntax:

```
while <condition> do
	statements
done
```

```
ankit@SecurityCrap:~$nano whileexample.sh
#!/bin/bash
i=0
while [ "$i" -le 5 ]
do
	echo "Line number $i"
	i=$(($i + 1))
done
exit 0
```

```
ankit@SecurityCrap:~$sh whileexample.sh
Line number 0
Line number 1
Line number 2
Line number 3
Line number 4
Line number 5
```

Since bash shell treats all variables as strings, make sure you follow that syntax to perform arithmetic operations. Another thing you need to watch out for is the usage of space, while declaring a variable do not put spaces before or after the equals sign.

Another thing to look out for in bash scripting is that its case sensitive, ankit, Ankit, ANKIT are all different for Unix.

Until

Similar to while, just the condition check is reversed. The program executes as long as the condition isn't true.

```
Syntax:
until <condition> do
        statements
done
```

```
ankit@SecurityCrap:~$nano untilexample.sh
#!/bin/bash
i=0
until [ $i -eq 5 ]
do
      echo "Line number $i"
      i=$(($i+1))
done
exit 0
```

```
ankit@SecurityCrap:~$sh untilexample.sh
Line number 0
Line number 1
Line number 2
Line number 3
Line number 4
```

Case

The case structure is best to use when a set of predefined input or values are to be accessed.

```
Syntax:
case <variable> in
        pattern1 | pattern2 | pattern3 ….) <statements>;;
        pattern4 | pattern 5 ..) <statements>;;
esac
```

```
ankit@SecurityCrap:~$nano caseexample.sh
#!/bin/bash
```

```
echo "Enter your Name:"
read name
echo "Enter your Gender:"
read sex

case $sex in
	M* | m* ) echo "Hello! Mr. $name";;
	F* | f* ) echo "Hello! Ms. $name";;
	* )	echo "Enter something valid for your gender, $name";;
esac
exit 0
```

```
ankit@SecurityCrap:~sh caseexample.sh
Enter your Name:
Ankit
Enter your Gender:
Male
Hello! Mr. Ankit
```

Functions

Declaring and using functions in bash-scripts gets easier than ever.

```
ankit@SecurityCrap:~nano function.sh
#!/bin/bash

func() {
	echo "This is the function executing..."
}

echo "Beginning of the script $0"
func
echo "End of the script $0"

exit 0
```

```
ankit@SecurityCrap:~sh function.sh
Beginning of the script function.sh
This is the function executing...
End of the script function.sh
```

I hope discussing these basics on shell-scripting will help you to get started with. There is a lot more to it, which can be said to be beyond the scope of this book. I might have to write an entire book based on bash-scripting, until then keep practicing these.

Email Encryption using pgp

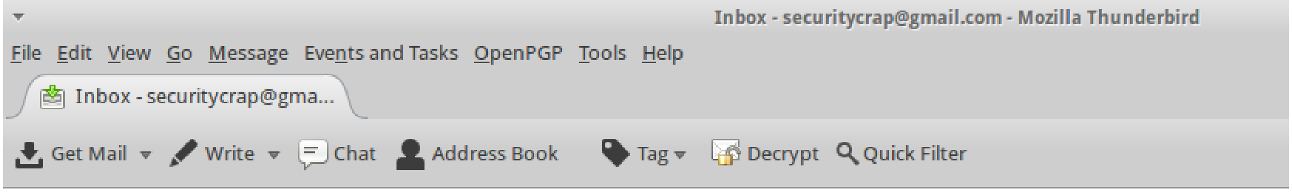
I have been a huge fan of thunderbird. I would suggest the readers to configure thunderbird, which is an open-source desktop email client designed by one of the most amazing Open-Source Communities out there - Mozilla. It comes by default in Ubuntu. Earlier versions of Ubuntu came with Evolution, but over the years it has been Thunderbird.

Fire up Thunderbird and start adding "Existing Email Account". If you have two-step authentication enabled in gmail, then kindly create application-specific password for Thunderbird, and then enter that password to transfer all mails to your local machine.

Then click on the "addons" option from "Tools" and search for "Enigmail". It adds OpenPGP message encryption and authentication to Thunderbird. What we are going to do here is create a set of public-private key. This mechanism is used to securely exchange emails. We will use a passphrase to create this asymmetric public-private keys, and publish our public-key. Any sender can use this public key to encrypt an email and even attachments to send to you, which only you can decrypt with your private key.

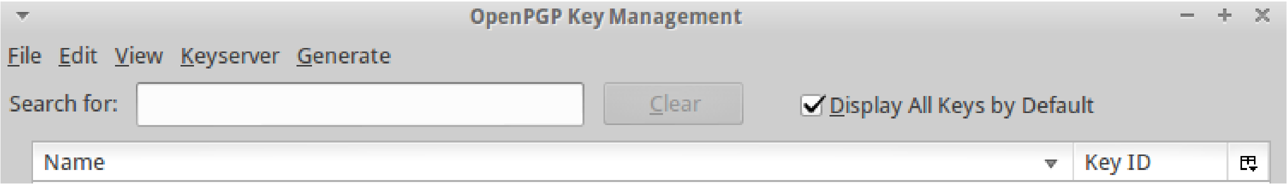
It might sound difficult, but enigmail makes it easy for you. Follow these steps:

On the menubar you will see OpenPGP.

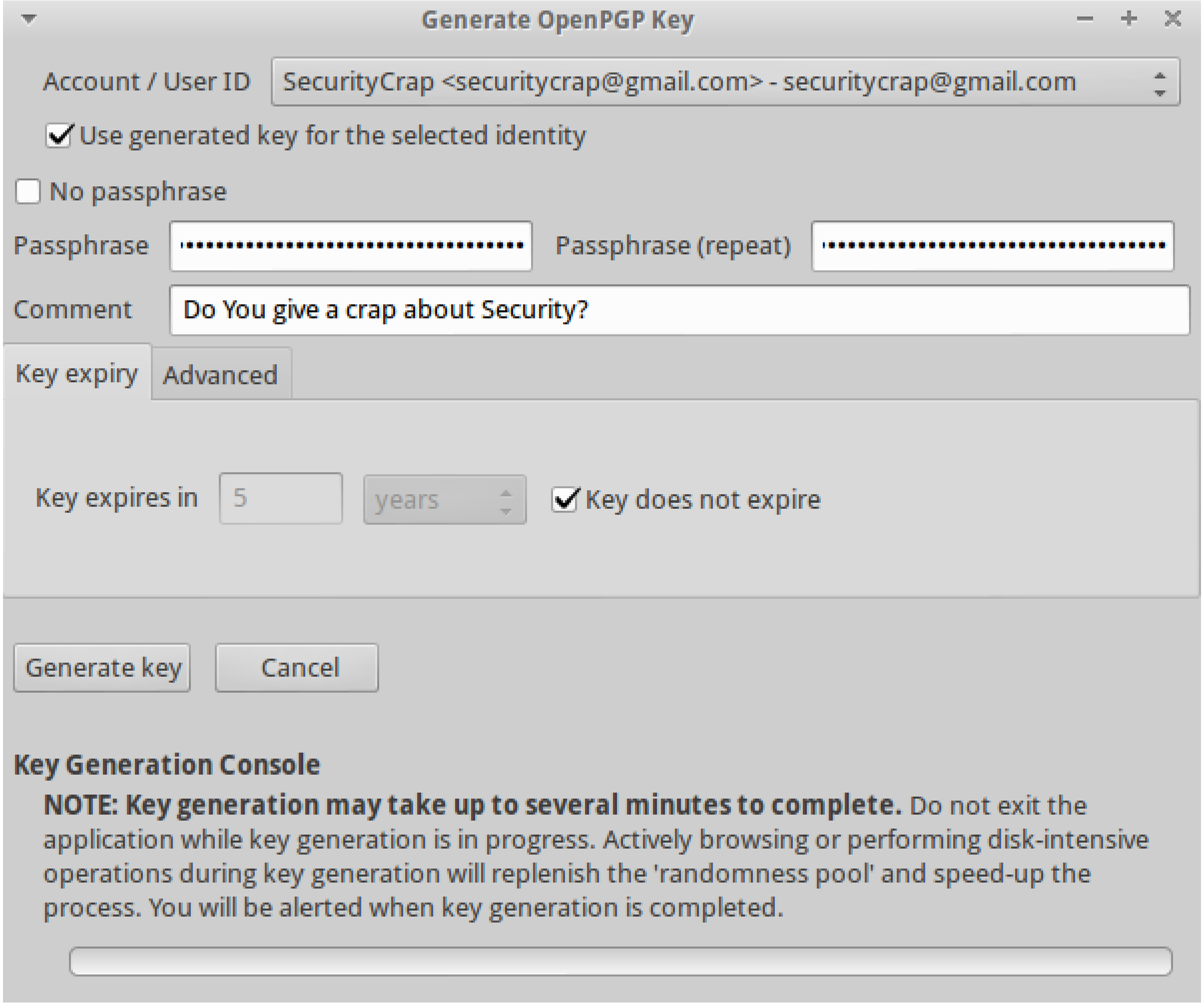

Inbox - securitycrap@gmail.com - Mozilla Thunderbird
File Edit View Go Message Events and Tasks OpenPGP Tools Help
Inbox - securitycrap@gma...
Get Mail
Write
Chat
Address Book
Tag
Decrypt
Quick Filter

Key Management

Click on that and then select "Key Management":


OpenPGP Key Management
File Edit View Keyserver Generate
Search for:
Clear
☑ Display All Keys by Default
Name
Key ID

Click on "Generate" and then select "New Key Pair"



Generate OpenPGP Key

Account / User ID: SecurityCrap <securitycrap@gmail.com> - securitycrap@gmail.com

☑ Use generated key for the selected identity

□ No passphrase

Passphrase: •••••••••••••••••••••••••••••••••••• Passphrase (repeat): ••••••••••••••••••••••••••••••••••••

Comment: Do You give a crap about Security?

Key expires in: 5 years ☑ Key does not expire

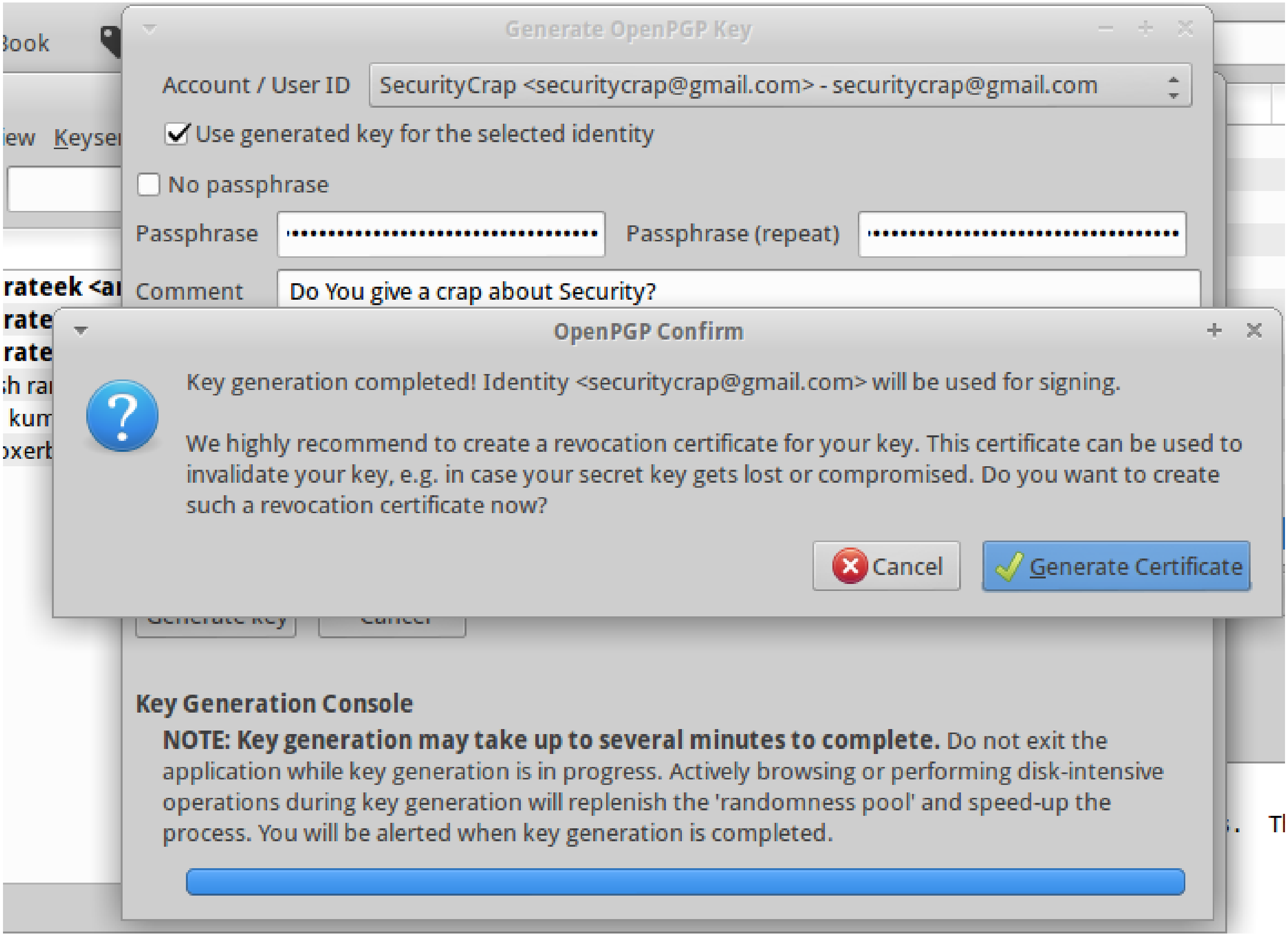
Key Generation Console

NOTE: Key generation may take up to several minutes to complete. Do not exit the application while key generation is in progress. Actively browsing or performing disk-intensive operations during key generation will replenish the 'randomness pool' and speed-up the process. You will be alerted when key generation is completed.

Generating a new key pair

Select your account user ID, and then choose a passphrase. Make sure the passphrase is strong enough. You can choose when the Key expires in that window.

After doing all that, click on “Generate Key”. After the Key has been created, a window will ask you to generate a Certificate. This is a revocation Certificate, this certificate is used to cancel out or invalidate your key in case your private key gets compromised.



Generate OpenPGP Key

Account / User ID: SecurityCrap <securitycrap@gmail.com> - securitycrap@gmail.com

☑ Use generated key for the selected identity

□ No passphrase

Passphrase: •••••••••••••••••••••••••••••••••••• Passphrase (repeat): ••••••••••••••••••••••••••••••••••••

Comment: Do You give a crap about Security?

OpenPGP Confirm

Key generation completed! Identity <securitycrap@gmail.com> will be used for signing.

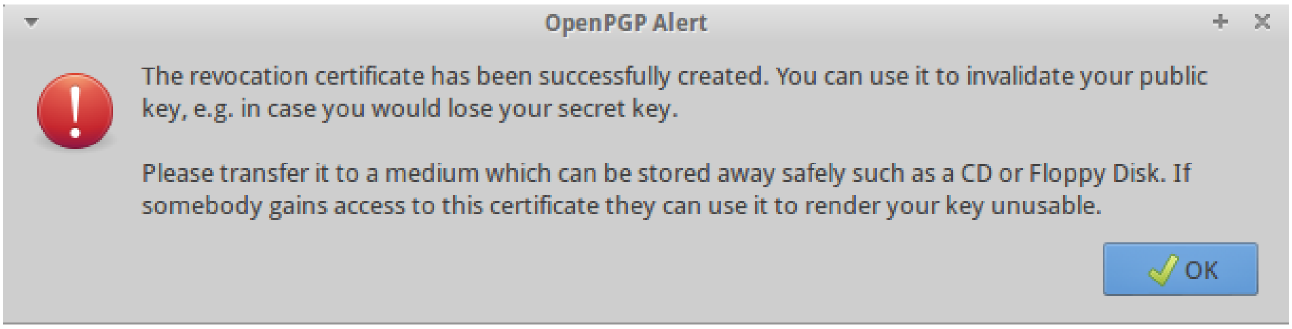
We highly recommend to create a revocation certificate for your key. This certificate can be used to invalidate your key, e.g. in case your secret key gets lost or compromised. Do you want to create such a revocation certificate now?

Key Generation Console

NOTE: Key generation may take up to several minutes to complete. Do not exit the application while key generation is in progress. Actively browsing or performing disk-intensive operations during key generation will replenish the 'randomness pool' and speed-up the process. You will be alerted when key generation is completed.

Creating revocation certificate

Choose a location to save that Certificate.

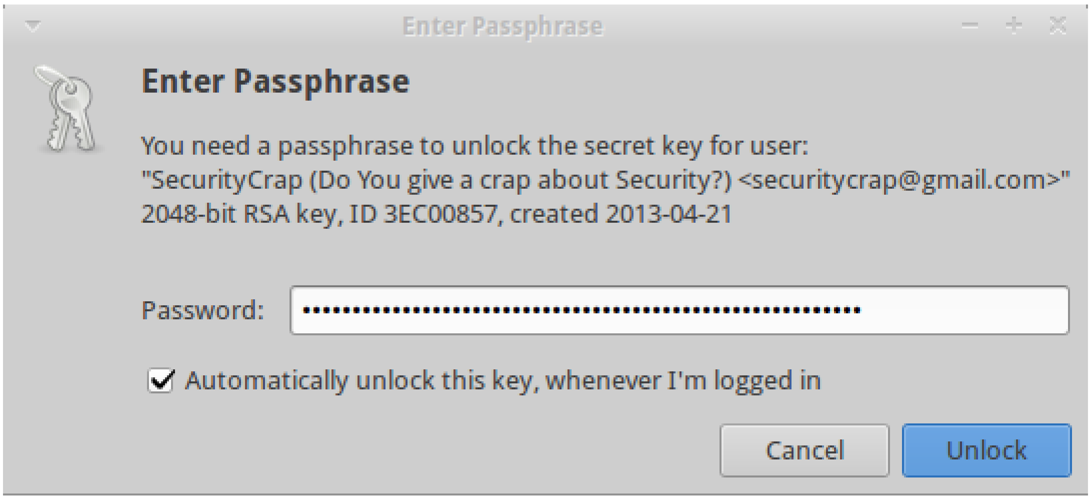


OpenPGP Alert

The revocation certificate has been successfully created. You can use it to invalidate your public key, e.g. in case you would lose your secret key.

Please transfer it to a medium which can be stored away safely such as a CD or Floppy Disk. If somebody gains access to this certificate they can use it to render your key unusable.

After that, you will be asked for the passphrase you just set. You can click on the checkbox that says
“Automatically unlock this key, whenever I'm logged in”.
Then every time you attach your public key while sending out an email, when you receive an encrypted email(encrypted using your public key of course) and add a signature to your emails, Enigmail will not ask you for your passphrase every time.



Enter Passphrase

You need a passphrase to unlock the secret key for user:
"SecurityCrap (Do You give a crap about Security?) <securitycrap@gmail.com>"
2048-bit RSA key, ID 3EC00857, created 2013-04-21

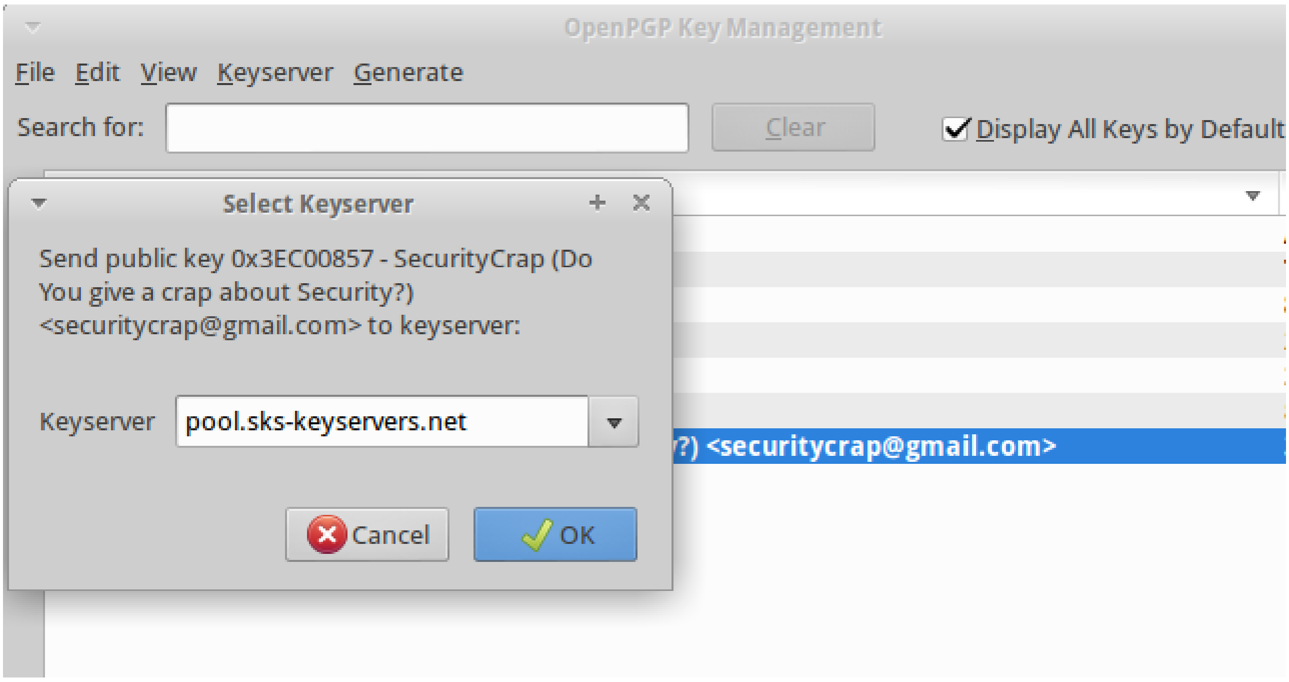
Password: ••

☑ Automatically unlock this key, whenever I'm logged in

With that, you have just successfully created the key-pair, and now you are good to go. When a friend sends you an encrypted email using your public key, even gmail won't be able to read its contents, forget any other low-class man-in-the-middle attacker.

Here let me go ahead and demonstrate sending/receiving encrypted emails. First, you need to publish your public key. You can do that by either publishing it on free pgp key servers or by attaching it as an attachment while sending emails. The receiver is less likely to check the pgp pool for your public key, but if you send them in an attachment they will at least know that you have a public key.

Go to the "Key Management" window and right click on your public key, select "Upload Public Keys to Key Server".



Search for:

☑ Display All Keys by Default

Send public key 0x3EC00857 - SecurityCrap (Do You give a crap about Security?) <securitycrap@gmail.com> to keyserver:

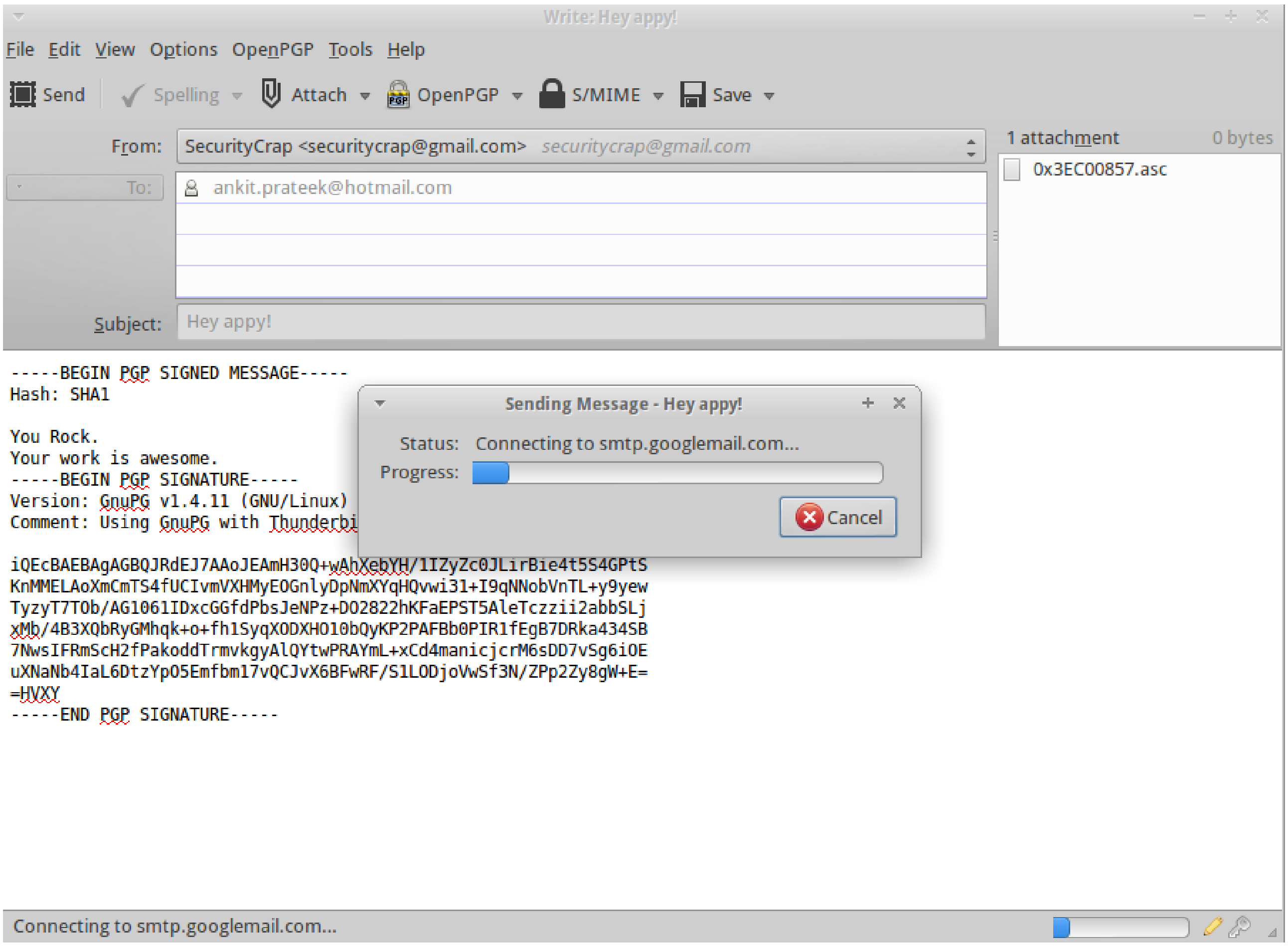
Keyserver: pool.sks-keyservers.net

Upload keys to a public key server

Choose any Keyserver and upload your key.

While sending an email, click on OpenPGP on the top and check "Attach Public Key".

The "Sign Message" is already checked.



From: SecurityCrap <securitycrap@gmail.com> securitycrap@gmail.com

To: ankit.prateek@hotmail.com

Subject: Hey appy!

1 attachment 0 bytes

0x3EC00857.asc

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

You Rock.
Your work is awesome.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.11 (GNU/Linux)
Comment: Using GnuPG with Thunderbi

iQEcBAEBAgAGBQJRdEJ7AAoJEAmH30Q+wAhXebYH/1IZyZc0JLirBie4t5S4GPtS
KnMMELAoXmCmTS4fUCIvmVXHMyEOGnlyDpNmXYqHQvwi31+I9qNNobVnTL+y9yew
TyzyT7TOb/AG1061IDxcGGfdPbsJeNPz+DO2822hKFaEPST5AleTczzii2abbSLj
xMb/4B3XQbRyGMhqk+o+fh1SyqXODXHO10bQyKP2PAFBb0PIR1fEgB7DRka434SB
7NwsIFRmScH2fPakoddTrmvkgyAlQYtwPRAYmL+xCd4manicjcrM6sDD7vSg6iOE
uXNaNb4IaL6DtzYp05Emfbm17vQCJvX6BFwRF/S1LODjoVwSf3N/ZPp2Zy8gW+E=
=HVXY
-----END PGP SIGNATURE-----
```

Sending Message - Hey appy!

Status: Connecting to smtp.googlemail.com...

Progress:

Connecting to smtp.googlemail.com...

Attach public key with email

You can see your email body changing from just a simple message to:

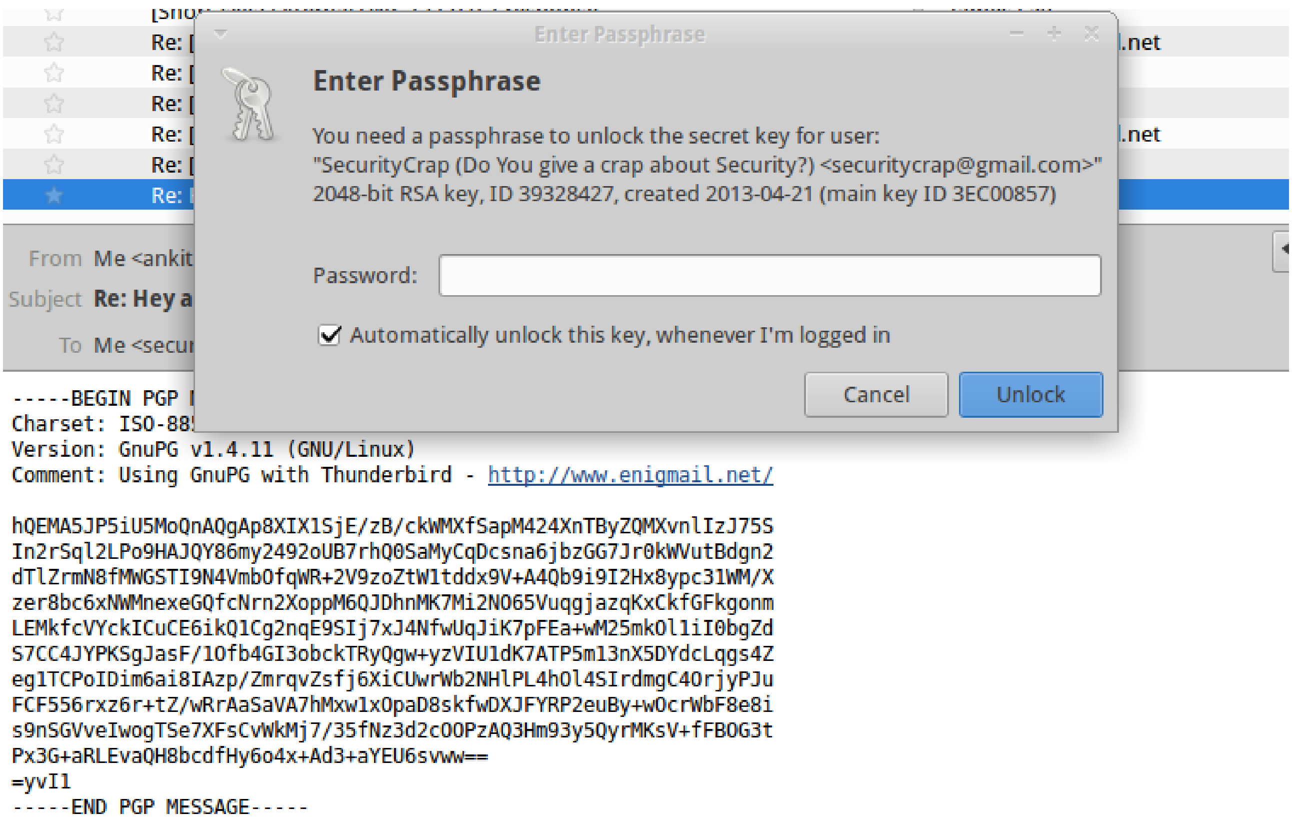
-------BEGIN PGP SIGNED MESSAGE---------

Then the message, and after that is your digital signature.

On the right your public key is sent as attachment.

The receiver can now use this public key to encrypt message and send to you.

Let me send a reply to that email, encrypted response using the public key just created. Here is how an encrypted email will look like.



Enter Passphrase

Enter Passphrase

You need a passphrase to unlock the secret key for user:
"SecurityCrap (Do You give a crap about Security?) <securitycrap@gmail.com>"
2048-bit RSA key, ID 39328427, created 2013-04-21 (main key ID 3EC00857)

Password:

☑ Automatically unlock this key, whenever I'm logged in

From Me <ankit

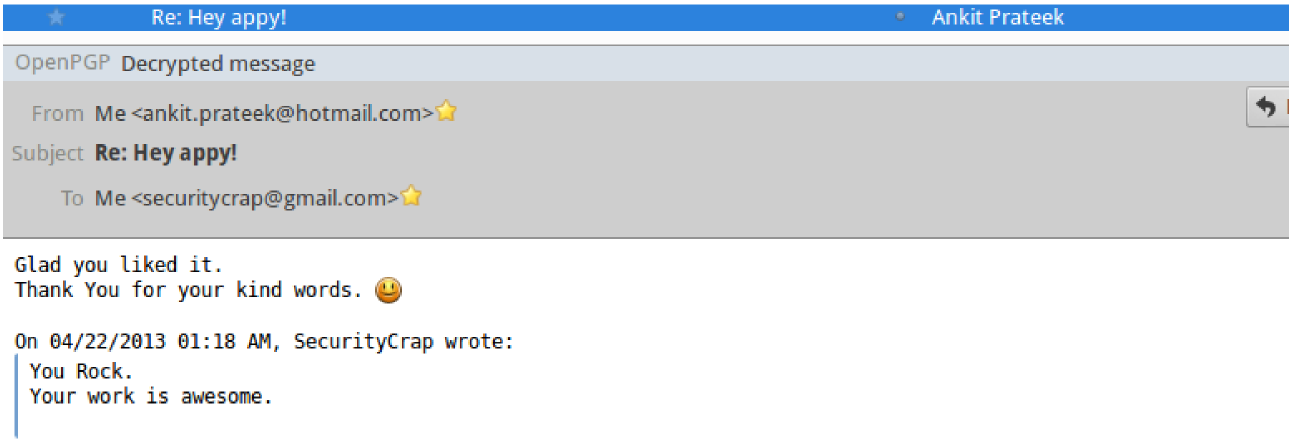
Subject **Re: Hey a**

To Me <secur

```
-----BEGIN PGP 
Charset: ISO-88
Version: GnuPG v1.4.11 (GNU/Linux)
Comment: Using GnuPG with Thunderbird - http://www.enigmail.net/

hQEMA5JP5iU5MoQnAQgAp8XIX1SjE/zB/ckWMXfSapM424XnTByZQMXvnlIzJ75S
In2rSql2LPo9HAJQY86my2492oUB7rhQ0SaMyCqDcsna6jbzGG7Jr0kWVutBdgn2
dTlZrmN8fMWGSTI9N4VmbOfqWR+2V9zoZtW1tddx9V+A4Qb9i9I2Hx8ypc31WM/X
zer8bc6xNWMnexeGQfcNrn2XoppM6QJDhnMK7Mi2NO65VuqgjazqKxCkfGFkgonm
LEMkfcVYckICuCE6ikQ1Cg2nqE9SIj7xJ4NfwUqJiK7pFEa+wM25mkOl1iI0bgZd
S7CC4JYPKSgJasF/1Ofb4GI3obckTRyQgw+yzVIU1dK7ATP5m13nX5DYdcLqgs4Z
eg1TCPoIDim6ai8IAzp/ZmrqvZsfj6XiCUwrWb2NHlPL4hOl4SIrdmgC40rjyPJu
FCF556rxz6r+tZ/wRrAaSaVA7hMxw1xOpaD8skfwDXJFYRP2euBy+wOcrWbF8e8i
s9nSGVveIwogTSe7XFsCvWkMj7/35fNz3d2cOOPzAQ3Hm93y5QyrMKsV+fFBOG3t
Px3G+aRLEvaQH8bcdfHy6o4x+Ad3+aYEU6svww==
=yvI1
-----END PGP MESSAGE-----
```

What encrypted mail looks like

The moment I click on that mail, I am prompted with a small window asking to enter passphrase to decrypt the email. And as I do that, here is what I see:



OpenPGP Decrypted message

From Me <ankit.prateek@hotmail.com>

Subject **Re: Hey appy!**

To Me <securitycrap@gmail.com>

```
Glad you liked it.
Thank You for your kind words. 🙂

On 04/22/2013 01:18 AM, SecurityCrap wrote:
> You Rock.
> Your work is awesome.
```

Decrypted message

The passphrase successfully decrypted the message with my private key and I can read the message in plain-text.

Enjoy exchanging private emails.