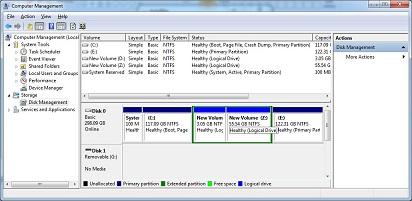
**Experiment 1: Installation of Linux (Ubuntu)**

**Aim :** To install Linux Ubuntu

**Theory :** Step 1:First Disk Partition is compulsory to allot memory to the linux operating System.

Type Disk management in the search bar

You will see the below image .



From the D: drive You can Free some space allot few GB’s to the linux drive

For that Follow the Given steps

1.Right Click on the D:drive ->Click on Shrink Volume -> Allot the Memory

Your free space is ready to use

**Step 2:** Take a pendrive make it as a bootable device

For making the device bootable delete all the contents in the pendrive and using a

Rufus Software make your device bootable.

Step 3:Restart Your Computer with the BIOS setup by pressing the F2 ,F9 or F10 the

It may vary according to the model .

**Note-: If You are using new model of DELL or HP you have to make some changes**

**In the BIOS set up Check for the Advance Boot Options in the BIOS setting .**

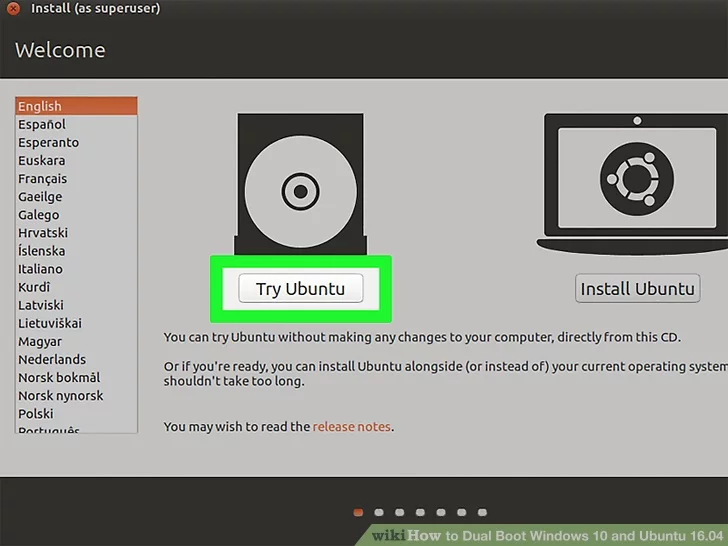
**There will be two modes available UEFI and Legacy mode. We Should disable the**

**UEFI mode .But for this we have to enable the Legacy mode . But again for that**

**Disable the Secure Boot .**

**Once you are done with this you are ready to boot with the bootable device**

**Step 3:**Now plugin your Bootable Device and Press F9 to install Ubuntu .



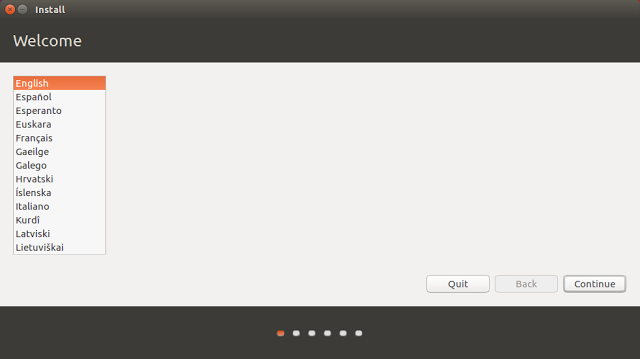
**Step 4:** After selecting the Bootable device You will get two option

1.Try Ubuntu

2.Install Ubuntu

Click on Install Ubuntu

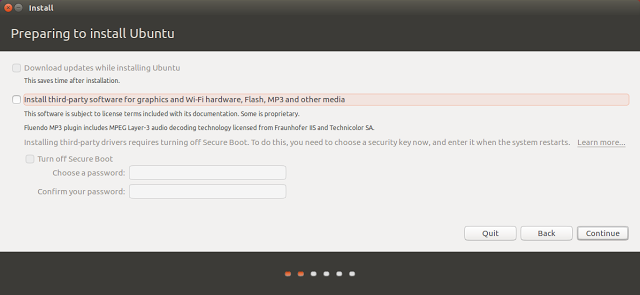
Then You will get the Following Page given below



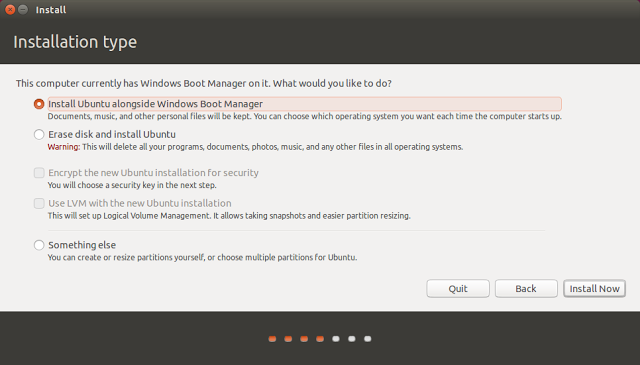
**Step 5:**This is the beginning of the installation process and you

can select the language which is used to help you through the process.

Choose your language and click “Continue”.



**Step 6: Without Choosing any of the option click “Continue”.**

****

**Step 7:** If you want to dual boot your machine then please select “something else”

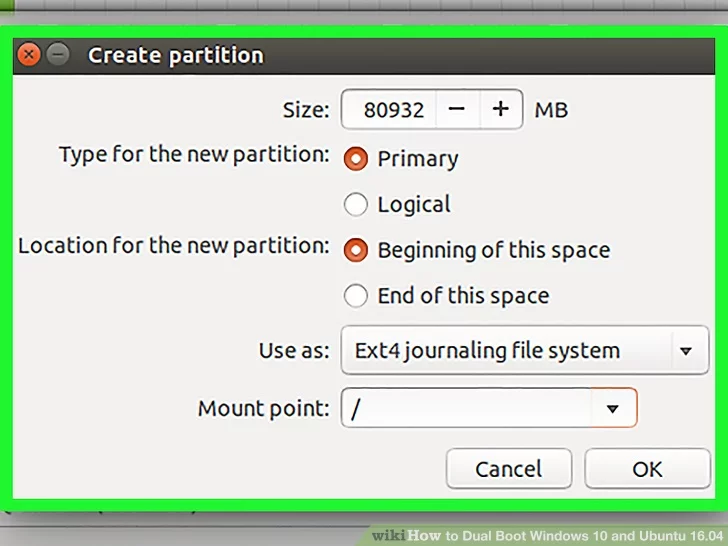
Or If you want o override windows you can select the first option .

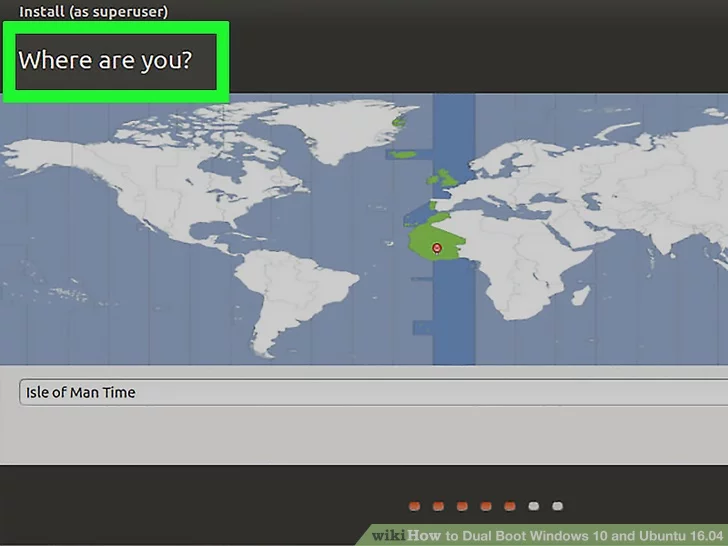
Step 8: You will be asked to allocate memory first the unallocated disk should be divided

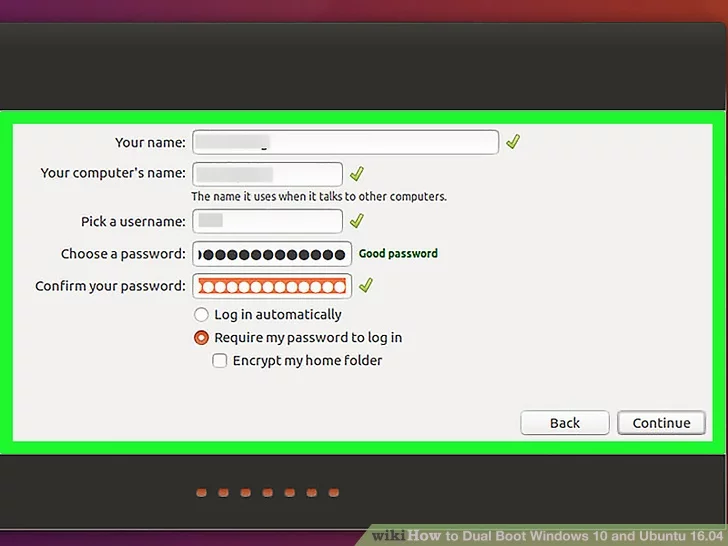
Into two parts Swap area which is logical device and should be allocated 2 times of the

Main memory. Remaining will be allocated to the primary memory .By setting the option

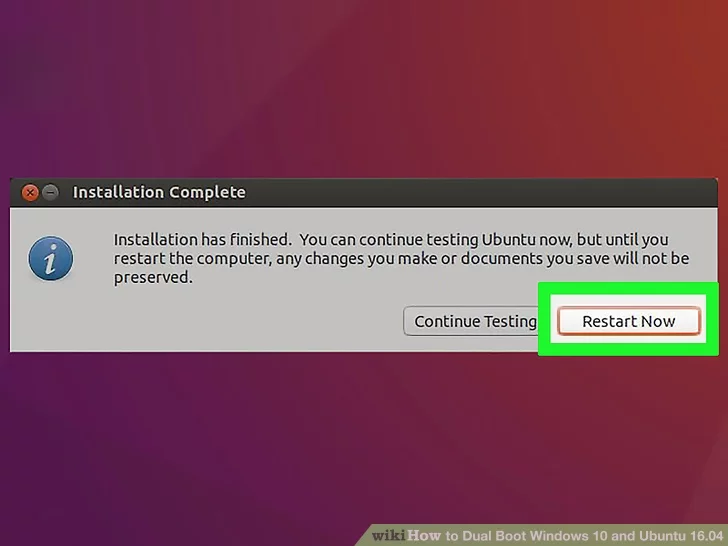
“/” You can click on continue.











Ubuntu installed successfully.

Our Next task is to bring windows as our primary OS

This can be done by using grub customizer

**Steps to install Grub customizer**

**1.** Open terminal from App Launcher or via Ctrl+Alt+T keys. When it opens, paste below command to add PPA.

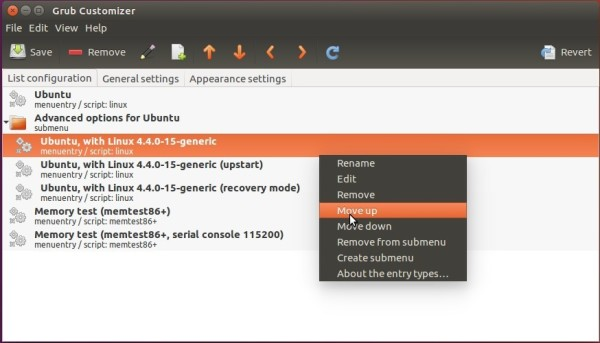
sudo add-apt-repository ppa:danielrichter2007/grub-customizer

**2.Type The Password if you have kept one**

Then update package lists and install the software:

sudo apt-get update  
  
sudo apt-get install grub-customizer

After installation type GRUB in the terminal and then you will see a penguin icon click on the penguin icon . You will see the below image:



By using the arrow keys bring windows in the top

**Booting Process**

### **Booting**

When a computer is turned on, **BIOS** finds the configured primary bootable device (usually the computer's hard disk) and loads and executes the initial bootstrap program from the master boot record (MBR). The MBR is the first sector of the hard disk, with zero as its offset (sectors counting starts at zero). For a long time, the size of a sector has been 512 bytes, but since 2009 there are hard disks available with a sector size of 4096 bytes, called Advance Format disks. As of October 2013, such hard disks are still accessed in 512-byte sectors, by utilizing the 512 emulation

The legacy MBR Partition table supports a maximum of four partitions and occupies 64 bytes. Together with the optional disk signatures (four bytes) and disk timestamp (six bytes), this leaves between 434 and 446 bytes available for the of a boot loader. Although such a small space can be sufficient for very simple boot loaders,[[7]](https://en.wikipedia.org/wiki/GNU_GRUB#cite_note-7) it is not big enough to contain a boot loader supporting complex and multiple file systems, menu-driven selection of boot choices, etc. Boot loaders with bigger footprints are thus split into pieces, where the smallest piece fits into and resides within the MBR, while larger piece(s) are stored in other locations (for example, into empty sectors between the MBR and the first partition) and invoked by the boot loader's MBR code.

Operating system [kernel](https://en.wikipedia.org/wiki/Kernel_(computing)) images are in most cases files residing on appropriate file systems, but the concept of a file system is unknown to the BIOS. Thus, in BIOS-based systems, the duty of a boot loader is to access the content of those files, so it can be loaded into the [RAM](https://en.wikipedia.org/wiki/RAM) and executed.

**BootLoaders**

**What is Bootloader?**

Bootloader is a piece of code that runs before any operating system is running.

Bootloader are used to boot other operating systems, usually each operating system has a set of bootloaders specific for it.

Bootloaders usually contain several ways to boot the OS kernel and also contain commands for debugging and/or modifying the kernel environment.

In this talk we will concentrate on Linux bootloaders.

Since it is usually the first software to run after power up or reset, it is highly processor and board specific.

**MBR**

A **master boot record** (**MBR**) is a special type of boot sector at the very beginning of partitioned computer mass storage devices like fixed disks or removable drives intended for use with IBMPC-Compatible systems and beyond. The concept of MBRs was publicly introduced in 1983 with PCDOS 2.0

The MBR holds the information on how the logical partitions, containing file systems, are organized on that medium. The MBR also contains executable code to function as a loader for the installed operating system—usually by passing control over to the loader's second stage, or in conjunction with each partition's volume boot record (VBR). This MBR code is usually referred to as a boot loader.

The organization of the partition table in the MBR limits the maximum addressable storage space of a disk to 2 TIB (232 × 512 bytes). Approaches to slightly raise this limit assuming 33-bit arithmetics or 4096-byte sectors are not officially supported, as they fatally break compatibility with existing boot loaders and most MBR-compliant operating systems and system tools, and can cause serious data corruption when used outside of narrowly controlled system environments. Therefore, the MBR-based partitioning scheme is in the process of being superseded by the GUID Partition table (GPT) scheme in new computers. A GPT can coexist with an MBR in order to provide some limited form of backward compatibility for older systems.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Firmware** | | **Multi-boot** | **File Systems** | | | | | **Notes** |
| **BIOS** | UEFI | **BTRFS** | **ext4** | **ReiserFS v3** | **VFAT** | **XFS** |
| **GRUB** | Yes | Yes | Yes | without zstd compression | Yes | Yes | Yes | Yes | On BIOS/GPT configuration requires a **BIOS boot Partition** |
| **system-md-boot** | No | Yes | Yes | No | No | No | Yes | No | Cannot launch binaries from partitions other than **ESP** |
| **Syslinux** | Yes | **Partial** | **Partial** | without: multi-device volumes, compression, encryption | without: 64bit feature, encryption | No | Yes | v4 on **MBR**only | No support for certain **file**systems features |
| **EFISTUB** | No | Yes | N/A | N/A | N/A | N/A | N/A | N/A |  |
| **rEFind** | No | Yes | Yes | without: encryption, zstd compression | without encryption | without tail-packing feature | Yes | No |  |
| **CLover** | emulates UEFI | Yes | Yes | No | Unknown | No | Yes | No | Main target audience is **Hackintosh** users. |
| **LILO** | Yes | No | Unknown | Unknown | Unknown | Unknown | Unknown | MBR only | **Deprecated**. Does not support  GUID Partition table |
| **GRUB**  **LEGACY** | Yes | No | Yes | No | No | Yes | Yes | v4 only | **Deprecated** Does not support **GUID**Partition table |

**GRUB**

**GNU GRUB** (short for **GNU GRand Unified Bootloader**) is a boot loader package from the GNU Projec. GRUB is the reference installation of the Free Software Foundation Specification which provides a user the choice to boot one of multiple operating systems installed on a computer or select a specific kernel configuration available on a particular operating system's partitions.

GNU GRUB was developed from a package called the Grand Unified Bootloader. It is predominantly used for Unix-like systems. The GNU operating systems uses GNU GRUB as its boot loader as do

most Linux distribution and the Solaris Operating systems on x86 systems, starting with the Solaris 10 1/06 release.

When booting from **32-bit ISO image**, it is said that **SYSLINUX** handles the boot process (the boot which shows beautiful splash screen, with progress dot animation).

But when booting from **64-bit ISO image**, **GRUB** is used instead (the black and white screen, clearly showing GNU GRUB version... on top of screen).

**SYSLINUX**

The Syslinux Project covers lightweight bootloaders for MS-DOS FAT filesystems (SYSLINUX), network booting (PXELINUX), bootable "El Torito" CD-ROMs (ISOLINUX), and Linux ext2/ext3/ext4 or btrfs filesystems (EXTLINUX) The project also includes MEMDISK a tool to boot legacy operating systems (such as DOS) from nontraditional media; it is usually used in conjunction with PXELINUX and ISOLINUX.

More info about SYSLINUX, PXELINUX, ISOLINUX, and EXTLINUX can be found on their respective pages; however, since the three have a lot in common, the common documentation is on the SYSLINUX page for now.

The Syslinux Project is now maintained in a public git repository. See Development for details.

There is a general mailing list that carries SYSLINUX announcements as well as discussion. If you have a problem it is a very good idea to browse the archives for information.

If you're interested in helping out with the SYSLINUX Project, please see the help wanted page, or just help out with the Wiki!

**CLOVER**

This is EFI-based bootloader for BIOS-based computers created as a replacement to EDK2/Duet bootloader

To compile it needs to place Clover sources into edk2/ folder.

**LILO**

**LILO** (**Linux Loader**) is a boot loader for LINUX and was the default boot loader for most LINUX distribution in the years after the popularity of loading. Today, many distributions use [GRUB](https://en.wikipedia.org/wiki/GNU_GRUB) as the default boot loader, but LILO and its variant ELILO are still in wide use. Further development of LILO was discontinued in December 2015 along with a request by Joachim Weidorn for potential developers.

**GRUB Legacy**

[GRUB Legacy](http://www.gnu.org/software/grub/grub-legacy.html) is a [multiboot](http://www.gnu.org/software/grub/manual/multiboot/) bootloader previously maintained by the [GNU Project](https://wiki.archlinux.org/index.php/GNU_Project). It was derived from GRUB, the GRand Unified Bootloader, which was originally designed and implemented by Erich Stefan Boleyn.

Briefly, the *bootloader* is the first software program that runs when a computer starts. It is responsible for loading and transferring control to the Linux kernel. The kernel, in turn, initializes the rest of the operating system.