Computer Hardware Essentials II

Prepared by
Pradeep Kumar P
Dept. of ICTS, Amrita School of Engineering, Amrita Vishwa Vidyapeetham ,
Coimbatore - 641112

BIOS and UEFI

BIOS

- **The basic input-output system (BIOS) is the first thing you see when you turn on** your computer.
- The BIOS is special software that interfaces the major hardware components of your computer with the operating system.
- It is usually stored on a Flash memory chip on the motherboard (which has its own battery), but sometimes the chip is another type of ROM.
- The BIOS also determines the compatibility of the computer and its flexibility in use. Although all BIOSs have the same function; all are not the same.

Users using the BIOS user interface can perform functions such as:

- Setting the system clock
- Enabling and disabling certain system components
- Hardware configuration
- Selecting boot drives
- Set password prompts for secured access to BIOS user interface function

- It is a firmware embedded on the chip on the computer's motherboard. BIOS firmware is pre-installed on the motherboard of a PC. It is a non-volatile firmware which means its settings won't disappear or change even after power off.

- The BIOS software has a number of different roles, but its most important role is to **load the operating system**. When you turn on your computer and the microprocessor tries to execute its first instruction, it has to get that instruction from somewhere.

- It cannot get it from the operating system because the operating system is located on a hard disk, and the microprocessor cannot get to it without some instructions that tell it how. The BIOS provides those **instructions**.

- In many PCs, this firmware also governs how the system board components interact, the chipset features that are used, even the amount of the microprocessor's time devoted to keeping memory working. The setup procedures in most new PCs are also held in the BIOS.

- The BIOS is both hardware and software. Like software, the BIOS is a set of instructions to the computer's microprocessor. Like hardware, however, these instructions are not evanescent; rather they are coded into the hard, worldly silicon of PROM, EPROM chips.

- The distinct parts of the BIOS operate separately and distinctly although the code for each is contained inside the same silicon chip. The BIOS operates like a set of small *terminate and stay-resident programs* that are always in memory. In this case, they are always in memory because we cannot get them out.

- IBM had envisioned that programs would never have to directly address hardware. Instead they would call up a software routine in the BIOS that has the addressing part of the instruction permanently set in its code.

- If a different hardware arrangement is used then, the address inside the routines would be changed to match the updated hardware.

- The same software could thus work with a wide variety of hardware designs, giving the designer and the manufacturer the flexibility to upgrade the entirety of the system hardware should the need arise.

- Modern PCs have BIOS stored in rewritable memory, permitting contents to be rewritten or replaced. Such content rewriting is called flashing and is executed through a special program provided by system manufacturers.

BIOS Purpose

- The design of any computer requires that many of the hardware elements of the machine be located at specific addresses within the range of input/output ports of the computer.

- Other computer components may have registers of their own that are used in their control. Because of the number of separate components inside any computer, the potential number of possible variations is limitless.

- Software that attempts to control any of this hardware must correctly reach out to these registers. As long as all computers are crafted exactly the same, with the same port used for exactly the same hardware with exactly the same registers, there should be no problem.

Usual sequence of BIOS

- Check the CMOS Setup for custom settings
- Load the interrupt handlers and device drivers
- Initialize registers and power management
- Perform the power-on self-test (POST)
- Display system settings
- Determine which devices are bootable
- Initiate the bootstrap sequence

BIOS Manufacturers:

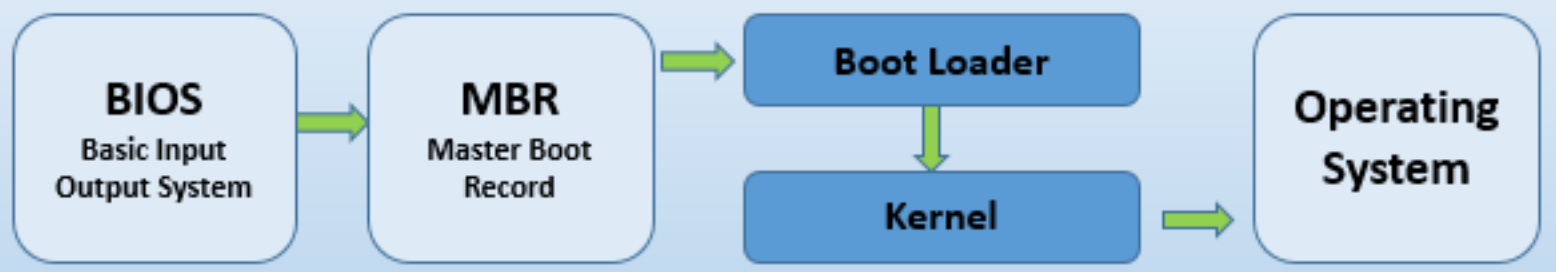
- American Mega Trends Inc. (AMI),
- Phoenix Technologies,
- Ali,
- Winbond.
- Award

Boot-Up Process

- A PC cannot do anything useful unless it is running its **operating system** – software that acts as a supervisor for all its software applications. It sets the rules for using memory, drives, and other hardware devices on the computer.

- Before a PC can run the operating system, it needs some way to load it from disk into RAM. The way to do this is with the **bootstrap** – a small amount of code that is executed on startup or system boot. The bootstrap is aptly named because it lets the PC do something entirely on its own 5, without any outside operating system.

BIOS Booting



Boot-Up Sequence

- The computer loads the **basic input/output system** (BIOS) from ROM. The BIOS provides the most basic information about storage devices, boot sequence, security, Plug and Play (auto device recognition) capability and a few other items.

- The BIOS triggers a test called a **power-on self-test** (POST) to make sure all the major components are functioning properly. You may hear your drives spin and see some LEDs flash, but the screen, at first, remains black.

- The BIOS has the CPU send signals over the system bus to be sure all of the basic components are functioning. The bus includes the electrical circuits printed on and into the motherboard, connecting all the components with each other.

- The POST tests the memory contained on the display adapter and the video signals that control the display. This is the first point you'll see something appear on your PC's monitor.

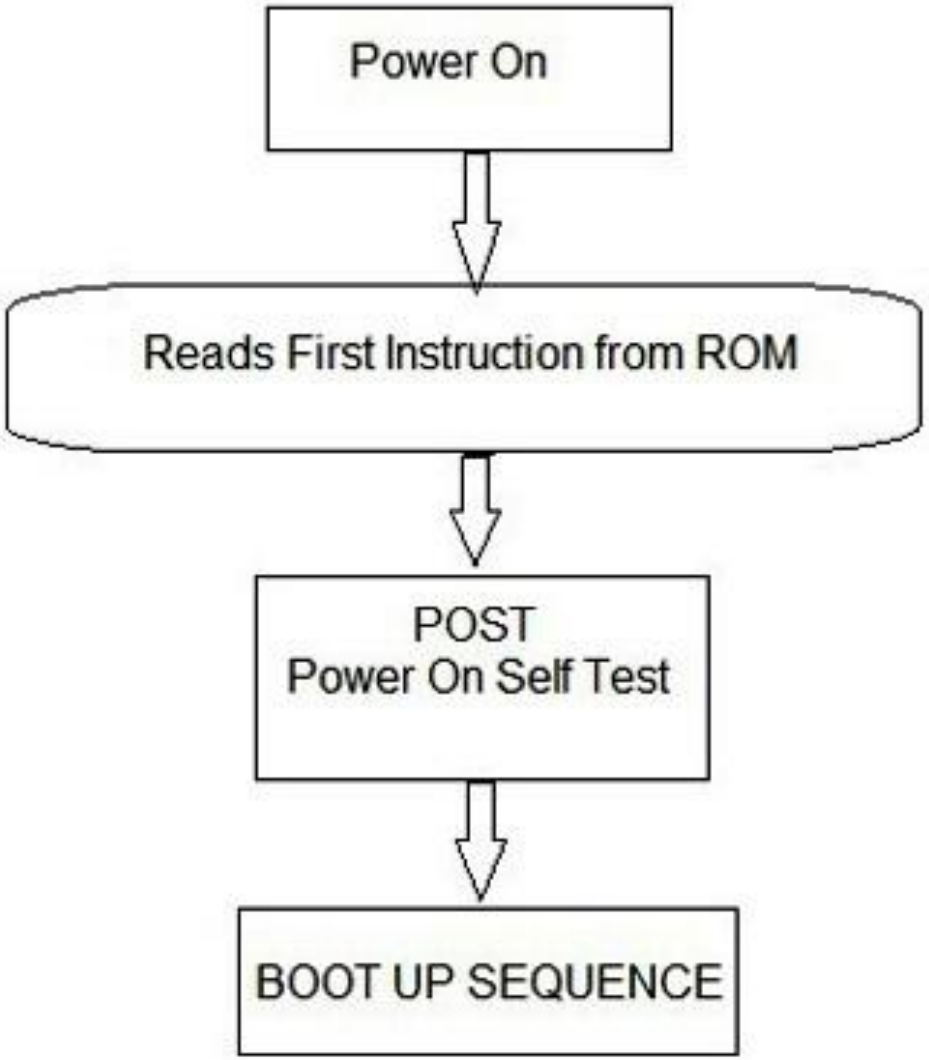
- During a **cold boot** the **memory controller** checks all of the memory addresses with a quick read/write operation to ensure that there are no errors in the memory chips. Read/write means that data is written to a bit and then read back from that bit. You should see some output to your screen - on some PCs you may see a running account of the amount of memory being checked.

- The computer **loads the operating system** (OS) from the hard drive into the system's RAM. That ends the POST and the BIOS transfers control to the operating system. Generally, the critical parts of the operating system - the **kernel** - are maintained in RAM as long as the computer is on. This allows the CPU to have immediate access to the operating system, which enhances the performance and functionality of the overall system.

POST-Power On Self Test

- In many cases, with IBM-compatible or PC computers, the POST is run by a computer's basic input/output system (BIOS).
- The initial tests, which are executed by the read-only memory (ROM) BIOS startup program, include reading configuration information stored in the complementary metal-oxide-semiconductor (CMOS) chip, dual inline package (DIP) switches and jumpers.
- This information is then compared to hardware devices, such as the CPU, memory, hard drive, disc drives and video card.
- Then, ROM BIOS assigns system resources as needed.
- These set up the environment required by the operating system (OS). After completion of these tests, POST generally alerts the OS with one or more beeps, depending on the system.

POST Sequence



- The BIOS issues an error message if hardware is not working properly or if it is not identified. The error message consists of text on the display screen or a series of coded beep sounds. Because POST is initiated prior to activating the video card, a display screen message is not typical.

- There are a variety of beep codes that are properly descriptive for troubleshooting the error. A beep code may indicate a parity error, base memory read/write (R/W) error, memory refresh timer error, display memory error, motherboard timer not functioning, cache memory failed or numerous other errors.

- Sometimes, an error stops the boot process until the error is corrected, and a device with an error is not allowed to run, ensuring safety. An error message can be basic. Sometimes a POST error can be drastic, such as when the motherboard does not detect a RAM component.

- POST is part of a devices pre-boot sequence. When POST is successfully finalized, bootstrapping is enabled. Bootstrapping starts the initialization of the OS.

- If a problem is found, it is reported with a code number on the monitor or as a coded series of beeps if an insufficient portion of the PC is functional to display anything on the monitor

- POST is the key to successful booting of the operating system. If the BIOS finds anything faulty on the computer, it will stop the booting and the message will be displayed on the screen.

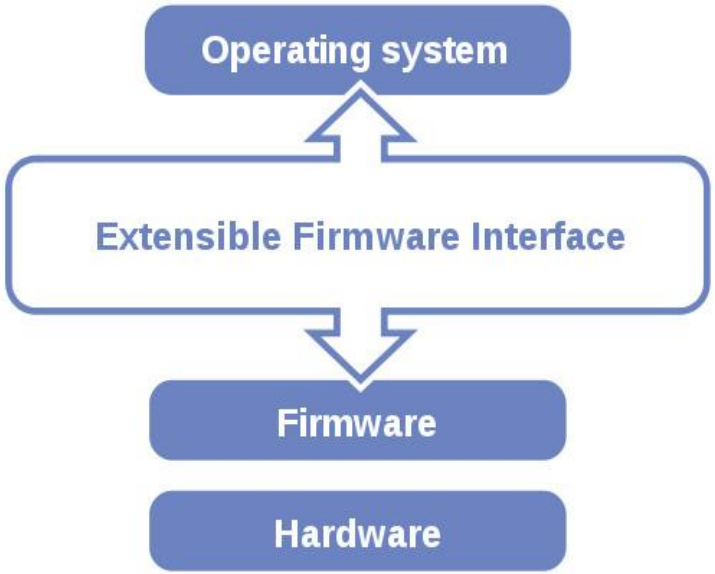
- If there is a problem before the display is activated, POST will notify you about the problem by giving a beep or a combination of beeps. These audio beeps are BIOS-dependent.

- Every BIOS manufacturer has its own set of beeps to convey the problem. The most common of all BIOS standards is the IBM BIOS and you can know the cause of the problem from Audio Beeps from your PC when it turns On and troubleshoot the computer accordingly.

Signal	Possible Cause
Continuous Beep	Keyboard Stuck
Repeating Short Beep	Power Supply Faulty
1 Long Beep followed by 1 short beep	Motherboard error
1 Long Beep , 2 short beeps	Display Card Faulty
1 short Beep, Blank screen	Check Display Cable or Display Card

UEFI - (Unified Extensible Firmware Interface)

- The UEFI (Unified Extensible Firmware Interface) specification defines an interface between operating systems and platform firmware. The interface consists of data tables that contain platform-related information, plus boot and runtime service calls that are available to the operating system and its loader. Together, these provide a standard, modern environment for securely booting an operating system and running pre-boot applications



Note: Some computer users use UEFI boot but still refer to it as the “BIOS”, which may confuse some people. Even if your PC uses the term “BIOS”, most modern PCs you buy today use UEFI firmware instead of a BIOS. To distinguish UEFI and BIOS, some also call UEFI firmware as UEFI BIOS, and BIOS is called Legacy BIOS or traditional BIOS.

- In addition to the services UEFI defines, there are various protocols/APIs to access various hardware and the boot devices in the system. The UEFI spec also defines a generic framework and can be adapted to any type of bus or device, knowing that computer hardware is constantly evolving

- There are 3 types of entities that can execute under UEFI environment

  - **Applications**:

    - Some examples of common UEFI applications include the UEFI shell, UEFI shell commands, flash utilities, and diagnostic utilities.

    - It is perfectly acceptable to invoke UEFI applications from inside other UEFI applications. .

    - Applications can reside inside firmware shipped on a system, or can be located/installed on storage media, such as a SSD/HDD, PCI card internal memory, or USB Key.

OS Loader:

- A special type of UEFI application, called an OS boot loader, provides the necessary initialization routines until the OS loader has set up enough of the OS infrastructure to be ready to assume complete ownership of the platform resources.

- OS Loaders, usually installed as part of the Operating System and usually located on the same storage media the Operating System is stored on, have the vital role of transitioning the system into Runtime mode, including coordinating virtual memory mapping to firmware code/data that will continue to be utilized during/after the OS has booted.

Drivers:

- UEFI drivers differ from UEFI applications in that the driver stays resident in memory unless an error is returned from the driver's entry point. The UEFI core firmware, the boot manager, or other UEFI applications may load drivers.

- The PCI spec refers to code on an external card or add-in device as an “Option ROM” or “oprom”, and therefore a UEFI driver may also be referred to as a “UEFI Option ROM” or “UEFI oprom” by some in the industry. UEFI drivers, like UEFI applications, can be found inside the system’s firmware, or on storage media, such as a SSD/HDD, PCI card internal memory, or USB Key.

UEFI have to offer over traditional BIOS

The primary goal of UEFI is to define an architecture that can scale with time, and offer a structured coding environment that allows easy enablement of newer technologies. Some of the distinguishing characteristics of UEFI, when compared to a traditional BIOS, are:

- **Abstraction for the OS.** The UEFI specification provides the interface between the platform firmware and the OS. The interfaces/API/protocols mark a clear boundary between the firmware and the OS.

- **Abstraction for devices and related code**. UEFI abstracts interfaces that make it possible to build code that works on a range of underlying hardware devices without having explicit knowledge of the specifics for each device in the range. This specification defines interfaces to platform capabilities including standard bus types such as PCI, USB, and SCSI. The list of supported bus types may grow over time, allowing code to utilize newer hardware through standard protocols without being rewritten.

- **Scalable platform environment.** The specification defines a complete solution for the firmware to describe all platform features and surface platform capabilities to the OS during the boot process. These definitions cover a range of the contemporary platform designs and the simple enough to be able to extend in the future.

- **OS Agnostic Rich Pre-Boot environment.** The UEFI spec defines extensible interfaces that enable creation of platform drivers. The UEFI drivers, analogous to OS drivers, provide support for new devices and may provide enhanced platform capabilities, such as firmware update, platform configuration, diagnostics and deployment services. The existence of networking, USB and file system capabilities adds to the richness of the pre-boot environment.

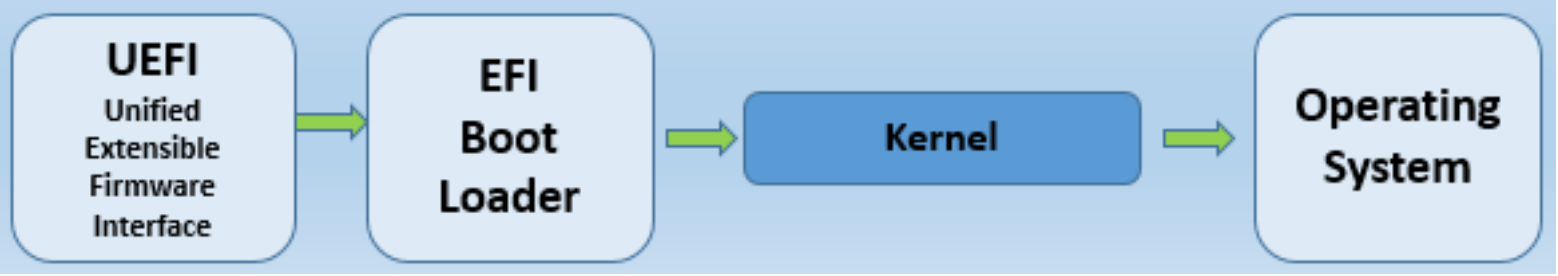
- **Consistent Configuration Infrastructure.** The UEFI spec defines a methodology of describing the platform configuring data in a standard way. The rendering of the data is left to the platform vendor. This allows UEFI to bring all the platform configurations like BIOS, Storage and Network options under a single setup application with a consistent navigation and look/feel.

- **GUID Partition Table.** The UEFI defines a new standard layout for the partition table known as GUID Partition Table (GPT). GPT provides a more flexible mechanism for partitioning disks than the older Master Boot Record (MBR) partitioning scheme that has been common to PCs. MBR disks support only four partition table entries and the partition size is limited to 2TB (2.20 × 1012 bytes). GPT scheme allows up to 128 primary partitions and can support partitions up to 9.4 Zettabytes (9.4 × 1021 bytes).

- There are some near-term limitations to 2 terabyte support due to device support, but once devices fully support GPT/UEFI, this will no longer be an issue (explained in more detail in the “Limitations” section).

- **Secure Boot.** The UEFI 2.2 (or later) specification brings security to the boot process by only loading the driver or OS loaders that are signed by a known/trusted digital signature. Secure boot keys are managed by the BIOS and OS. Secure boot can also be placed in a "Custom" mode, where additional public keys can be added by the platform administrator to allow execution of custom code or restriction of code that may be trusted by some, but not by the platform’s owner.

UEFI Booting



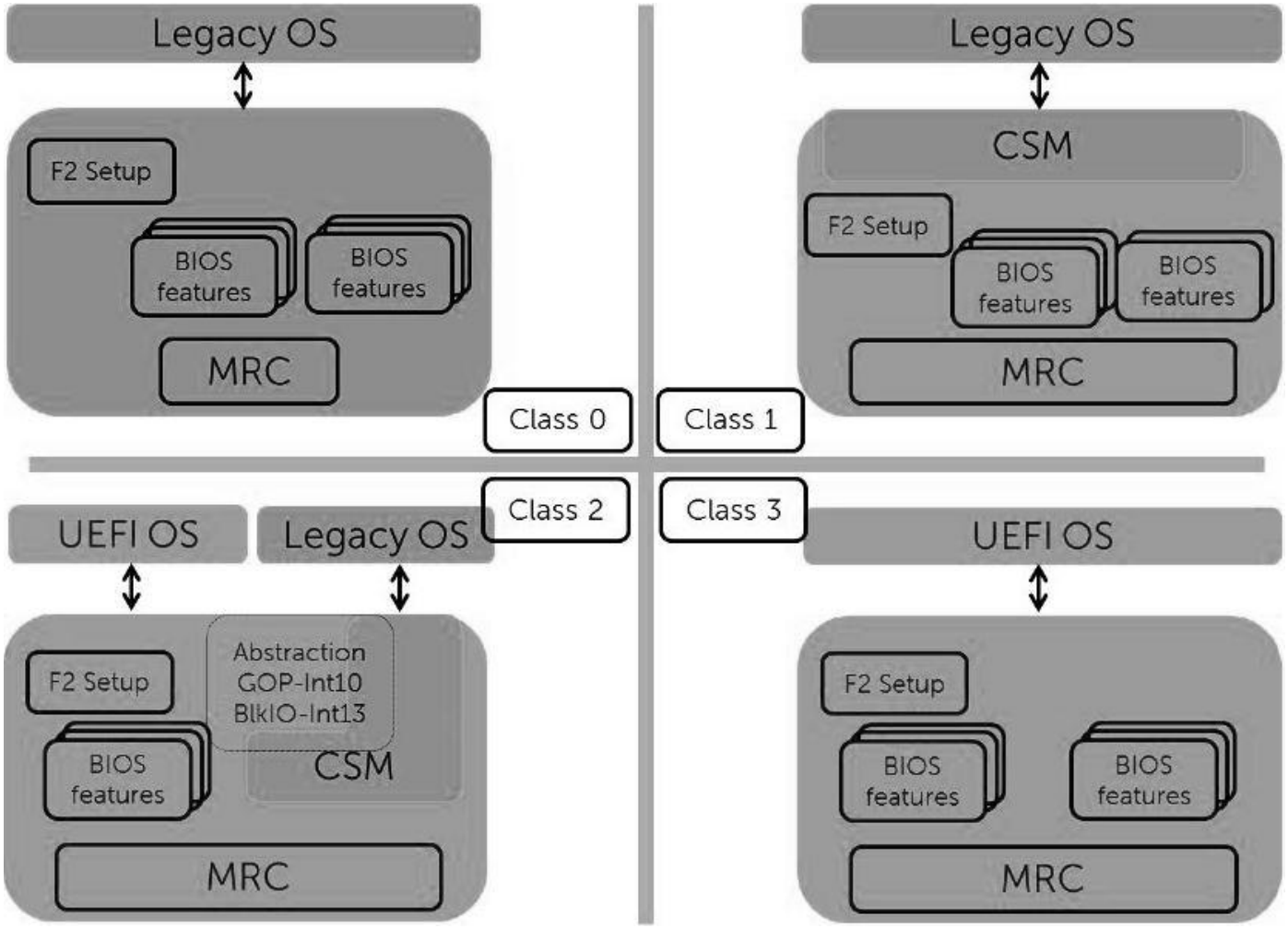
Distinguish between UEFI and Legacy

	Legacy BIOS	UEFI Firmware
Programming Language	Assembly	C
Processors Supported	Intel Architecture	Intel Architecture, Itanium, and ARM officially supported by UEFI Spec
Processor Mode Used	Mostly 16-bit, Single CPU, Single Threaded	Native (64-bit or32-bit), Single CPU, Single Threaded
Expansion Card Firmware	Legacy Option ROMs	UEFI Drivers or UEFI Option ROMs
Provided Services	Interrupts	Protocols
Video Support	Int10h & VBIOS	Graphics Output Protocol (GOP)

	Legacy BIOS	UEFI Firmware
Storage Support	Int13h, Master Boot Record (MBR) Partitioning	Block IO Protocol, GUID Partition Table (GPT) and Master Boot Record (MBR) Partitioning
Peripheral & Feature Setup	F2 Setup, Ctrl-M, Ctrl-A No Industry Standard	UEFI Human Interface Infrastructure (HII) Protocol as Industry Standard
OS Boot Loader	Int19h loads 16-bit boot sector in MBR One boot loader per device	UEFI loads boot loader executable file(s) per priority/ordering defined by UEFI Spec. Multiple boot loader files, unique names/paths, can co-exist on the same partition/device.
OS Handoff	No clear definition	Exit BootServices() function defined by UEFI Spec.

UEFI Platform Classification

- For better understanding the technology context and UEFI adaption progression the platforms can be classified in 4 classes.



UEFI Platform Classification

- **Class 0:** Non UEFI platforms, the set of platforms based on traditional legacy BIOS. These platforms are not UEFI aware.
- **Class 1:** In the earlier days of EFI/UEFI when all the leading OSs were not EFI/UEFI aware a special Compatibility Support Module (CSM) was used to present the traditional BIOS like interface. These platforms only booted to traditional, legacy OSs,
- **Class 2:** These platforms came about when EFI was adapted as UEFI industry standard and OS started adding support for UEFI. These platforms support booting using the traditional method of int19h, where in BIOS loads the boot sector and hands of execution to the boot loader, as well as loading a UEFI boot loader application. Majority of platforms shipping today are Class 2 platforms.
- **Class 3:** These platforms support booting only using the UEFI defined method of loading the boot loader application from a specific location. Class 3 platforms do not sport a Compatibility Support Module. In fact any class 2 platform with CSM turned off functions like a Class 3 platform.

UEFI Boot vs Traditional BIOS

The Memory Factor: BIOS can boot up from drives having memory of 2.1TB or less and with 3TB drives becoming very common a computer with traditional BIOS won't be able to boot up. This is because of how the Master Boot Record(MBR) of legacy BIOS functions as MBR uses 32-bit entries and limits you to maximum of 4 memory partitions.

UEFI on the other side uses GUID partition table (GPT) which uses 64-bit entries and can have infinite partitions however Windows limits to '128' partitions. Also it supports drives having memory greater than 2TB with theoretical value of maximum support of 9.4 zettabytes.

Performance: As UEFI has more addressable space than BIOS it allows the system to boot faster with faster initializing of hardware for your operating system.

Security: The biggest advantage of using UEFI over BIOS is that it has security which is not provided by the BIOS. Secure Boot is a characteristic feature of UEFI which ensures that no malware tampers with the boot process and ensuring that the operating system is clean as a whistle. In Windows this system ensures that no pirated copies of boot loaders have been used.

Booting- BIOS UEFI PXE

Legacy BIOS Boot Mode and UEFI Boot Mode

- Your system is equipped with UEFI BIOS, which is based on the Unified Extensible Firmware Interface (UEFI) specification.

- UEFI BIOS offers advantages over legacy versions of BIOS, but the way it boots is not compatible with some operating systems, and it might not be the best choice for some configurations. For this reason, the system can be configured to boot in **Legacy BIOS Boot Mode** or **UEFI Boot Mode**. **Legacy BIOS Boot Mode is the default.**

- Normally, you set the boot mode only once, before installing the operating system. Once you have installed the operating system, if you change the boot mode, you cannot boot the operating system.

When to Select Legacy or UEFI BIOS Boot Mode

When the option is available to choose between Legacy BIOS Boot Mode or UEFI Boot Mode, the advantages of UEFI Boot Mode include:

- Boots faster.
- Avoids legacy option ROM address constraints.
- Supports operating system boot partitions greater than 2 terabytes (2 TB). For more information about limitations for supported operating systems, refer to your server product notes.
- Integrates PCIe device configuration utilities with BIOS Setup Utility menus.
- Displays bootable operating system images in the boot list as labeled entities. For example, it displays Windows boot manager labels instead of raw device labels.
- Provides efficient power and system management.
- Includes robust reliability and fault management.
- Uses UEFI drivers.

Choose Legacy BIOS Boot Mode:

- If your operating system does not support booting in UEFI Boot Mode
- To allow HBAs and Express Module devices to use option ROM

Legacy Boot

- Legacy Boot is the boot process used by BIOS firmware.
- The firmware maintains a list of installed storage devices that may be bootable (Floppy Disk Drives, Hard Disk Drives, Optical Disk Drives, Tape Drives, etc...) and enumerates them in a configurable order of priority.
- One the POST procedure has completed, the firmware loads the first sector of each of the storage targets into memory and scans it for a valid Master Boot Record (MBR).
- If a valid MBR is found, the firmware passes execution to the boot loader code found in the MBR which allows the user to select a partition to boot from.
- If one is not found, it proceeds to the next device in the boot order.

UEFI Boot

- UEFI boot is the boot process used by UEFI firmware.
- The firmware maintains a list of valid boot volumes called EFI Service Partitions. During the POST procedure the UEFI firmware scans all of the bootable storage devices that are connected to the system for a valid GUID Partition Table (GPT).
- Unlike a MBR, a GPT does not contain a boot loader.
- The firmware itself scans the GPTs to find an EFI Service Partition to boot from.
- If no EFI bootable partition is found, the firmware can fall back on the Legacy Boot method.

Note: UEFI boot is more desirable.

Secure Boot

- UEFI has a firmware validation process, called secure boot. Secure boot defines how platform firmware manages security certificates, validation of firmware, and a definition of the interface (protocol) between firmware and the operating system.
- Microsoft's platform integrity architecture creates a root of trust with platform firmware using UEFI secure boot and certificates stored in firmware.
- A growing trend in the evolution of malware exploits is targeting the boot path as a preferred attack vector. This class of attack has been difficult to guard against, since antimalware products can be disabled by malicious software that prevents them from loading entirely.
- With Windows 8's secured boot architecture and its establishment of a root of trust, the customer is protected from malicious code executing in the boot path by ensuring that only signed, certified "known good" code and boot loaders can execute before the operating system itself loads.
- In most PCs today, the pre-operating system environment is vulnerable to attacks by redirecting the boot loader handoff to possible malicious loaders. These loaders would remain undetected to operating system security measures and antimalware software.

Legacy BIOS boot path

Windows 8 addresses this vulnerability with UEFI secure boot, and using policy present in firmware along with certificates to ensure that only properly signed and authenticated components are allowed to execute.

Secure boot path with UEFI

Preboot Execution Environment (PXE)

- Preboot execution environment (PXE), pronounced as "pixie," allows computers to boot up remotely through a network interface. PXE enables a client machine to boot from a server independent of the hard disks and installed operating system.

- PXE was introduced as a component in the Wired for Management (WfM) framework by Intel in 1999. Intel's WfM has now been superseded by Active Management Technology, but PXE is still a valuable tool for many network administrators around the world.

- This term is also known as pre-execution environment.

- Network booting is generally applied in a diskless environment using routers and centrally managed computers, also known as thin clients. Centralized computing environments provide reduced maintenance costs, enhanced security and enhanced control over the system's workstations.

- PXE code is generally delivered with computer machines on a ROM chip or a boot disk to enable remote boot and configuration. The process makes use of network protocols such as User Datagram Protocol (UDP), Trivial File Transfer Protocol (TFTP), Internet Protocol (IP) and Dynamic Host Configuration Protocol (DHCP).

- These protocols have been selected because they are easily implemented in the client's NIC firmware, resulting in standardized small-footprint PXE ROMs.

- Standardization, small size of PXE firmware images and their low use of resources are some of the primary design goals, allowing the client side of the PXE standard to be identically implemented on a wide variety of systems, ranging from powerful client computers to resource-limited single-board computers (SBC) and system on a chip (SoC) computers.

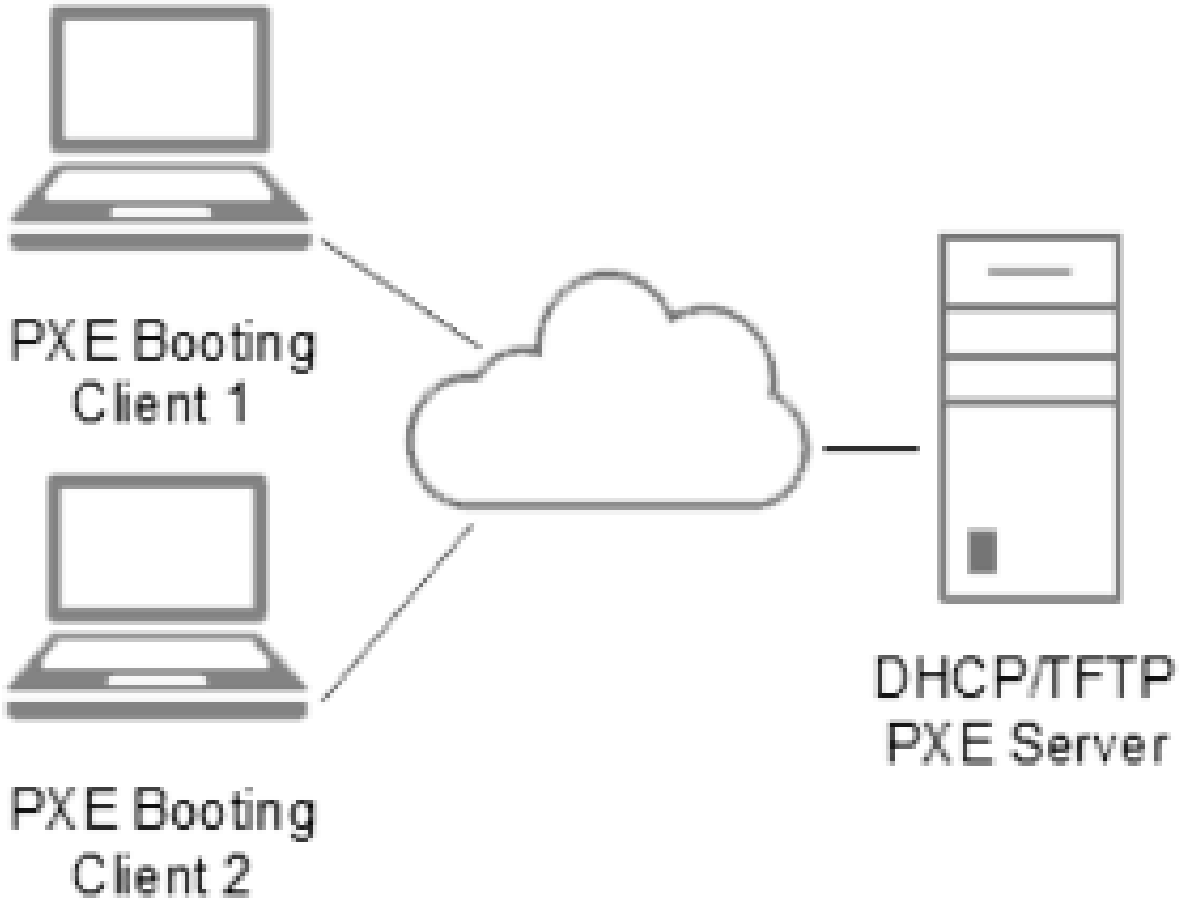
- Some of the key advantages of PXE are:
  - The client machine or workstation does not require a storage device or operating system.
  - Network extension and the addition of new client computers is made easier because PXE is vendor-independent.
  - Maintenance is simplified because most tasks are performed remotely.
  - Centralized data storage provides information security.

PXE Boot Process

When the client initiates a PXE boot (by traditionally pressing F12) however the process is changed slightly:

- The client sends out a DHCP broadcast and states that it needs to PXE boot
- The DHCP server picks up this broadcast and replies with a suggested IP address to use. If the server has the information on how to PXE boot, that information is included in it's reply
- The client then replies to the server and uses the provided address
- Then the client contacts the PXE boot server (traditional a WDS server or SCCM server) and requests the bootfile that it received from the DHCP server
- The file is then loaded and launched on the client Typically Option 66 or Option 67 are set within your DHCP scope options or DHCP Helpers are configured within your router for the above process to work. Option 66 specifics which server to contact and 67 is the name of the file to request.

Basic PXE Boot

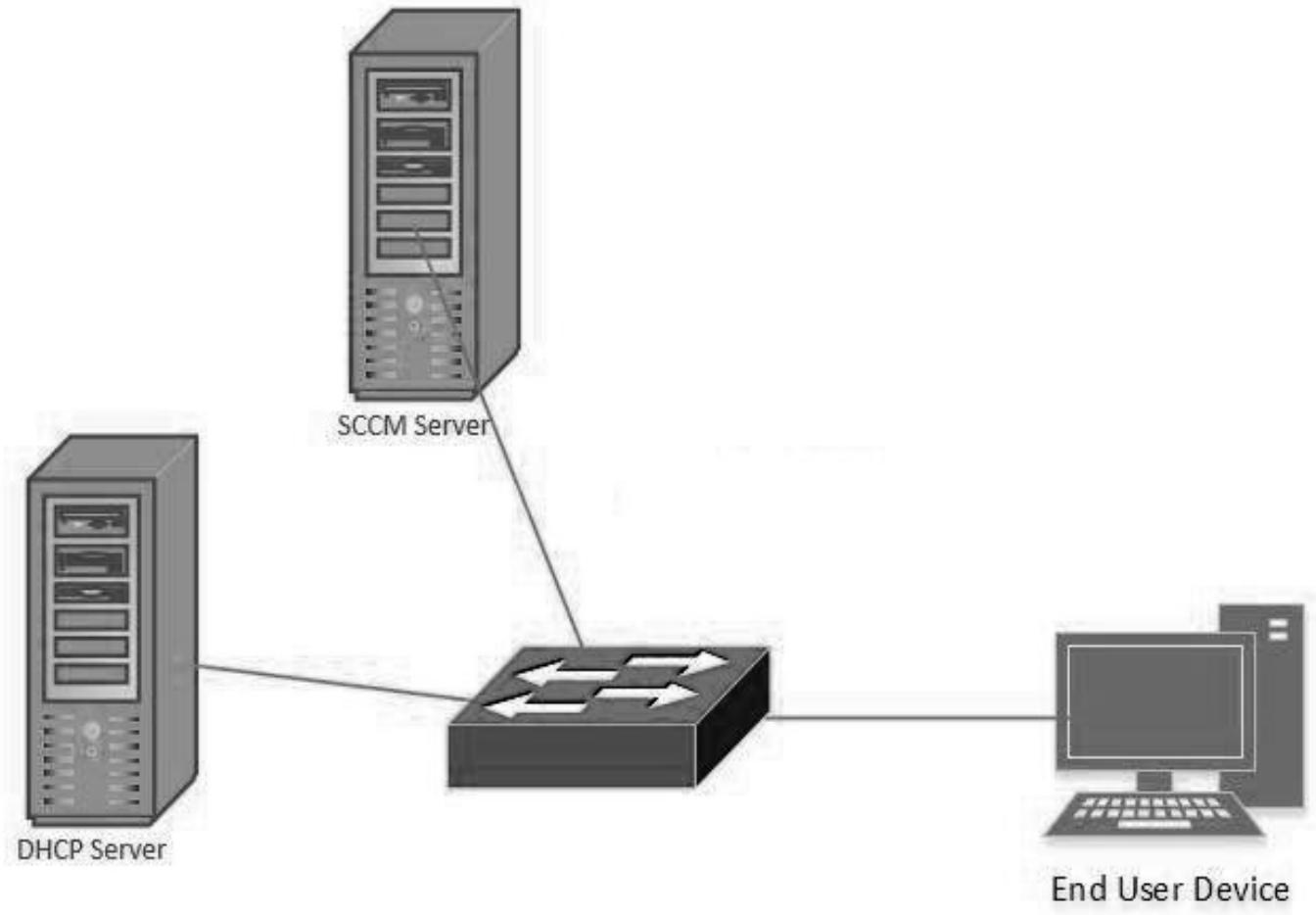


Dynamic PXE Boot

Another method of providing the boot information to a client is to have a service listen for the DHCP request. Configuration Manager provides dynamic PXE boot using the Windows Deployment Service (WDS).

- The client connects to the network and sends out a DHCP broadcast
- The DHCP server picks up this broadcast and replies with a suggested IP address to use.
- The WDS service also replies back to the client with the necessary information it needs to PXE boot
- The client replies to the DHCP server and uses the suggested address
- The client also contacts the PXE boot server and requests the bootfile specified in the boot information it was sent

Dynamic PXE Boot



PXE-Booting Steps

Step 1

Power on or restart your computer. Press the hotkey to enter the BIOS. Common keys are "F2" and "Del."

Step 2

Navigate to the network card settings with the keyboard. This is usually located under "Integrated Peripherals," though it may have its own separate section, depending on the motherboard.

Step 3

Toggle the "LAN BOOT ROM" option to "Enabled." Your motherboard may refer to this simply as "PXE."

Step 4

Navigate to the boot settings with they keyboard. Change the boot order so that the network is placed ahead of the internal hard drive. Most motherboards feature this setting under the "Advanced BIOS Features" or "Boot Settings." The boot devices are listed in preferred order. Use the arrow keys to navigate to the first boot device; then press the "Enter" key to change the device to "Network" or "LAN."

Press "F10" to save the settings and exit the BIOS. The computer restarts and boots from the network if a PXE server is present on the LAN.

Hard Disk Drive Partitioning

Partition

The **partitioning** of a hard drive occurs after the drive has been Physically formatted but before it is logically formatted. It involves creating areas on the disk where data will not be mixed.

To install different Operating Systems that do not use the same file system. There will, therefore, be at least as many partitions as there are operating systems using different file systems.

If you are using just one operating system, a single partition of the full size of the disk is sufficient. That is unless you want create several partitions so as to have, for example, several drives on which data is kept separate.

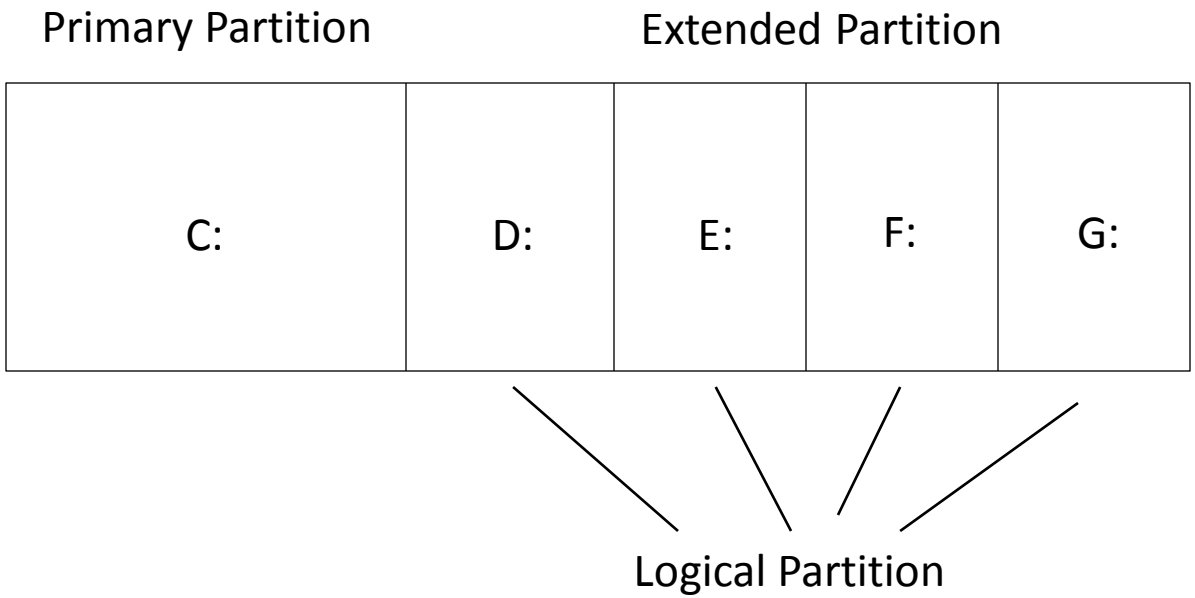
There are three types of partitions:

Primary partitions, **extended partitions** and **logical drives**.

- A disk may contain up to four primary partitions (only one of which can be active), or three primary partitions and one extended partition.

- In the extended partition, the user can create logical drives

The disk contains one primary partition and one extended partition made up of three logical drives:



For DOS systems (DOS, Windows 9x), only the primary partition is bootable and is, therefore, the only one on which the operating system can be started.

Using Multiple Partitions

- There are three types of partitions:

primary partitions, extended partitions and **logical drives.**

- A disk may contain up to four primary partitions or three primary partitions and one extended partition.
- In the extended partition, the user can create logical drives.

Primary Partition

- A primary partition must be logically formatted and have a file system appropriate to the operating system installed on it.
- If you have several primary partitions on your disk, only one will be active and visible at a time, depending on the operating system with which you started the computer.
- By choosing which operating to load at start-up, you determine which partition will be visible.
- The **active partition** is the partition from which one of the operating systems was loaded when the computer was started up.
- The partitions, other than the one from which you started, will then be hidden, which will prevent their data from being accessible.
- The data on a primary partition are therefore only accessible from the operating system installed on that partition.

Extended Partition

- Extended partitions were developed to overcome the limit of four primary partitions, as you can create as many logical drives as you want in them.
- At least one logical drive is required in an extended partition, as you cannot store data in them directly.
- Many machines are formatted with one large partition using up all available space on the drive.
- This is not, however, the most advantageous solution in terms of performance and capacity.
- The solution is to create several partitions, which will allow you to install several operating systems on your disk, save disk space, increase file security, and organize your data more easily

Master Boot Record

- The **boot sector** (called the **Master Boot Record** or **MBR**) is the first sector of a hard drive (cylinder 0, head 0, sector 1). It contains the **main partition table** and the code, called the **boot loader**, which, when loaded into memory, will allow the system to boot up.
- After it is loaded into memory, this program will determine from which system partition to boot, and will start the program (called the **bootstrap**), which will start up the operating system present on that partition.

- This disk sector also contains all of the information concerning the hard drive .This sector is, therefore, the most important one on the hard drive and is also used by the BIOS setup to recognize the hard drive. In other words, without it, your hard drive is useless, which makes it a favorite target for viruses.

File Systems

- It is important to differentiate between the **FAT file system** and the **file allocation table (FAT)**.

Operating System	Associated File System
Dos	FAT16
Windows 95	FAT16
Windows 98	FAT32
Windows NT	NTFS
Windows XP	NTFS/FAT32
Windows 2000	NTFS/FAT32
Windows 7	NTFS
Windows 8	NTFS
Windows 10	NTFS
OS/2	HPFS
Linux	Linux ext2,ext3,ext4

FAT (File Allocation Table)

- FAT file systems are characterized by the use of a file allocation table and clusters (or blocks).
- Clusters are the smallest unit of storage in a FAT file system. A cluster actually represents a fixed number of disk sectors.
- The FAT (**File Allocation Table**) is the heart of the file system. It is located in sector 2 of cylinder 0, head 1 (and is duplicated in another sector as a precaution in the event of an accident). This table records the numbers of the clusters that are used and where the files are located in the clusters.
- The FAT file system supports disks or partitions up to a maximum size of 2 GB but only allows at most 65,536 clusters. So, whatever the size of the partition or disk, there must be enough sectors per cluster so that the entire disk space can be contained in these 65,525 clusters. As a result, the larger the disk (or partition), the greater the number of sectors per cluster.

FAT (File Allocation Table) Cont..

- The FAT file system uses a **root directory** (represented on the operating systems that use this type of file system by the symbol C:\ ) , which must be located at a specific location on the hard drive.

- This root directory stores information on the sub-directories and files that it contains. For a file, it will store the file name, the file size, the date and time the file was last modified, the file attributes, and the cluster number at which the file starts.

OS Installation Windows 7

Step 1: Boot From the Windows 7 DVD or USB Device

To begin the Windows 7 clean install process, you'll need to boot from the Windows 7 USB or DVD if you're using a Windows 7 DVD, or boot from a USB device if your Windows 7 installation files are located on a flash drive or other external USB drive.

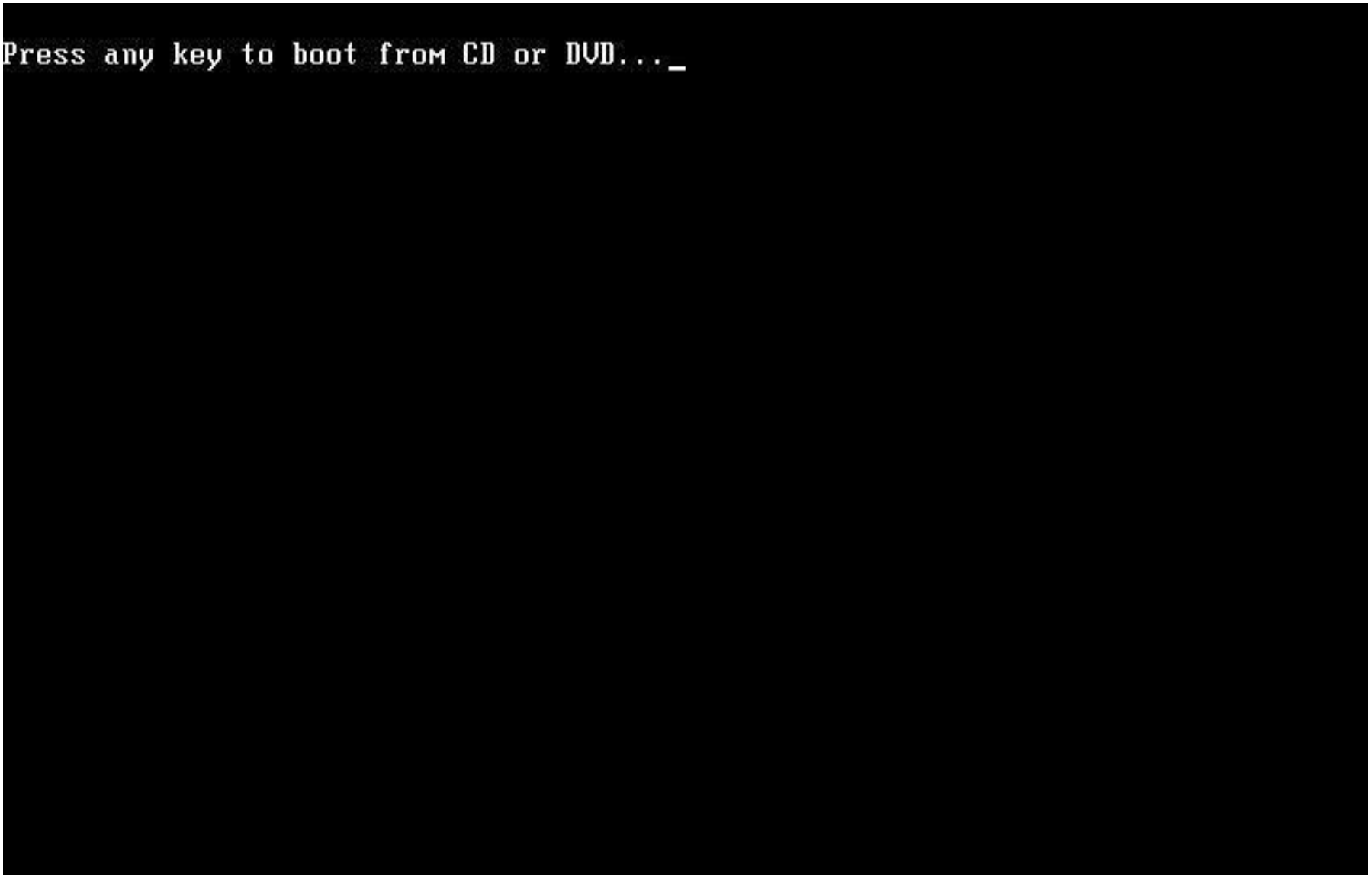
Tip: See my Windows Installation FAQ if you have Windows 7 as an ISO image that you need on a flash drive or disc, or a Windows 7 DVD you need on a flash drive.

Restart your computer with the Windows 7 DVD in your optical drive, or with the properly configured Windows 7 USB flash drive plugged in.

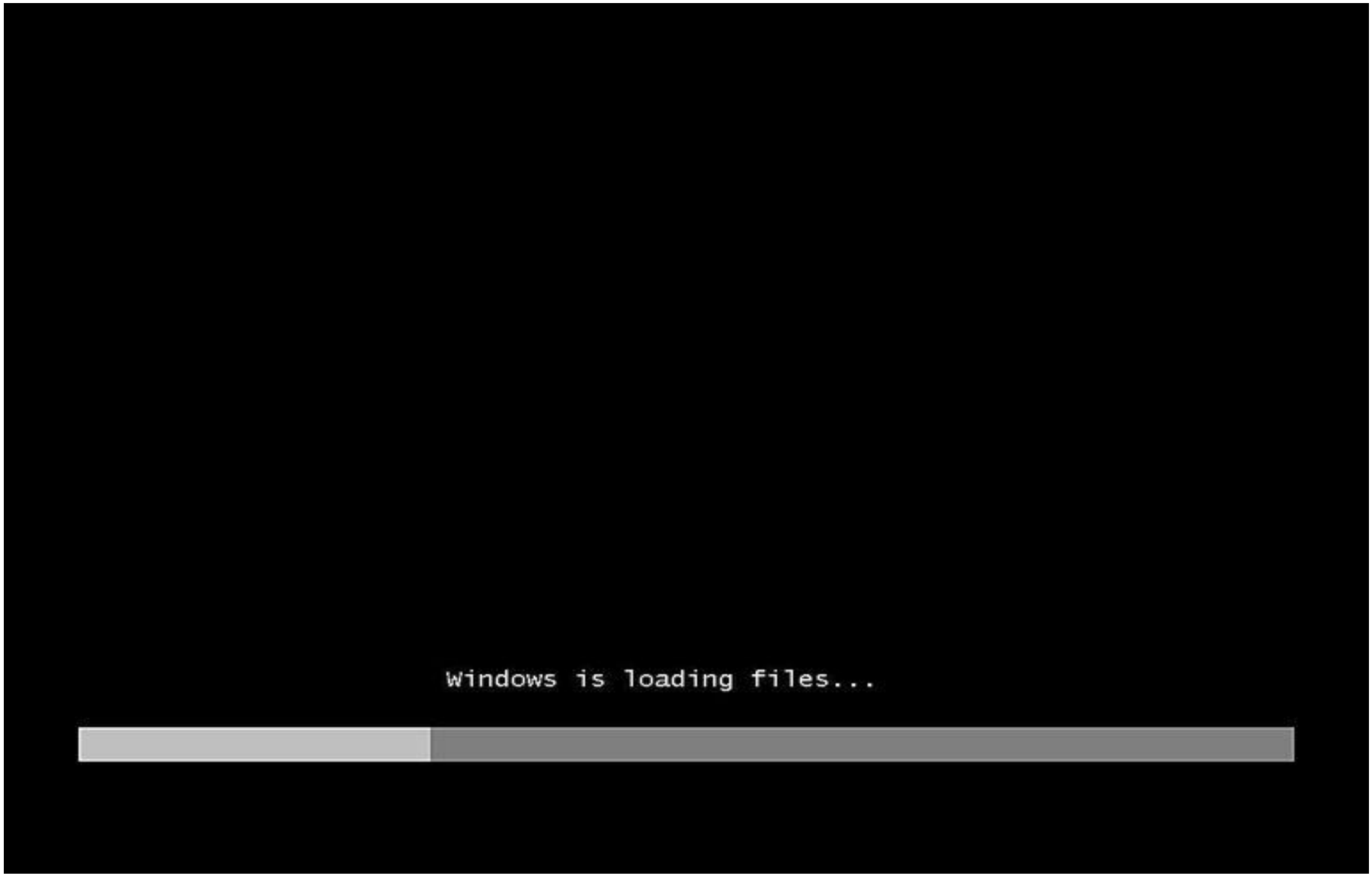
Watch for a Press any key to boot from CD or DVD... message similar to the one shown in the screenshot above. If you're booting from a flash drive, the message may be phrased differently, like Press any key to boot from external device....

Press a key to force the computer to boot from the Windows 7 DVD or USB storage device. If you do not press a key, your computer will attempt to boot to the next device in the boot order, which is probably your hard drive. If this happens, chances are your current operating system will boot.

Step 1: Contd..,

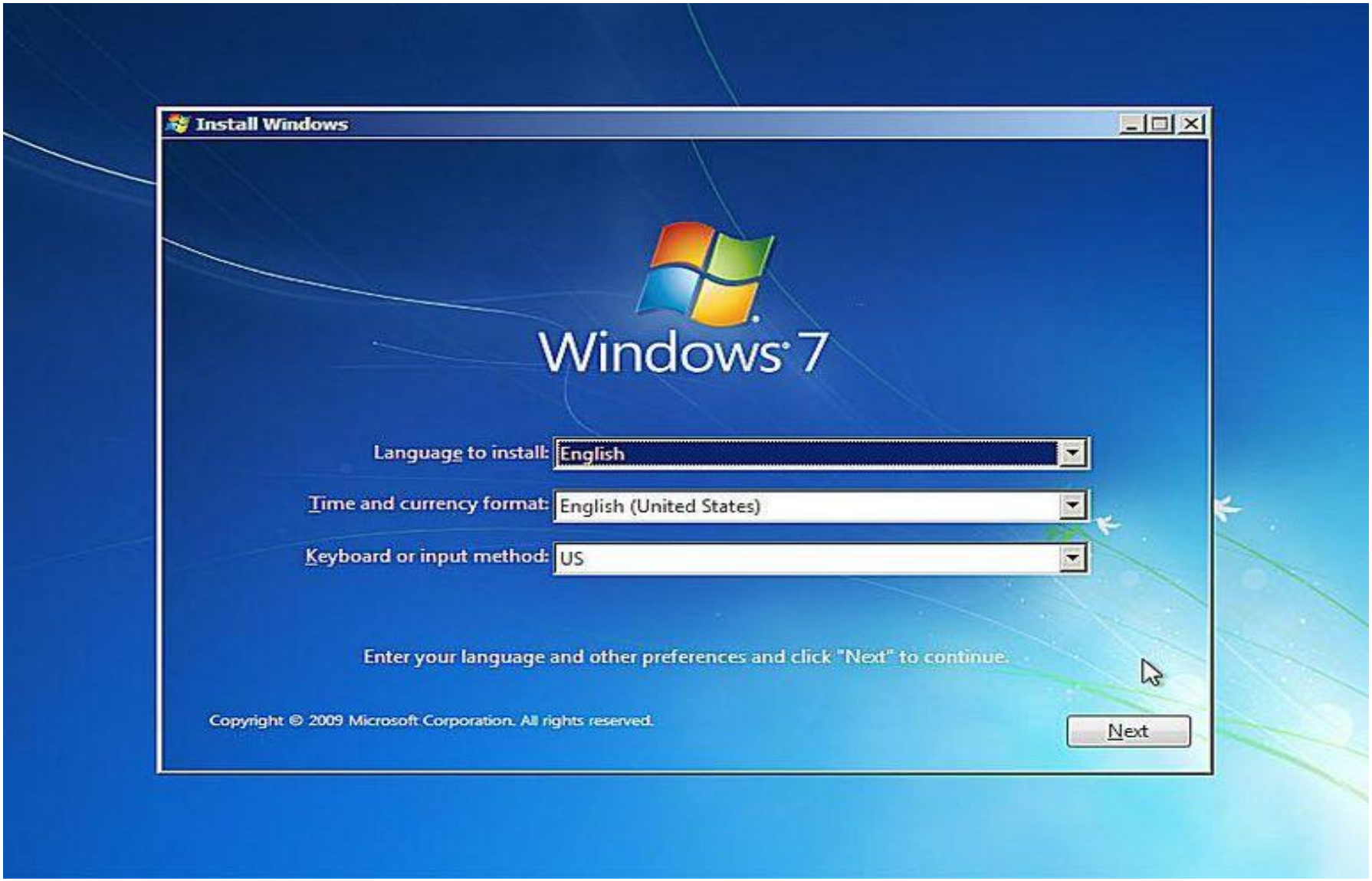


Step 2: Wait for Windows 7 Installation Files to Load

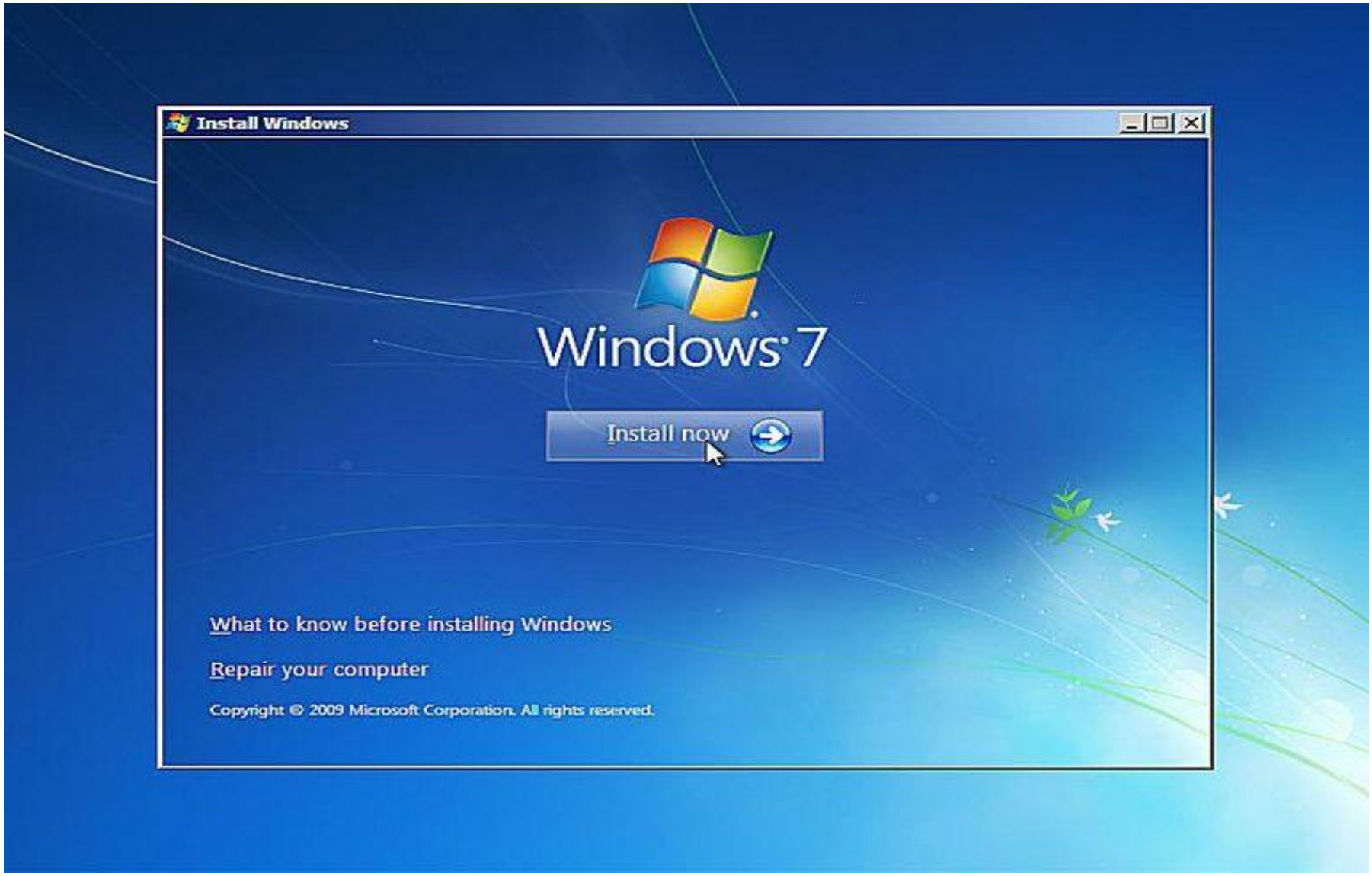


Step 3: Wait for Windows 7 Setup to Finish Loading

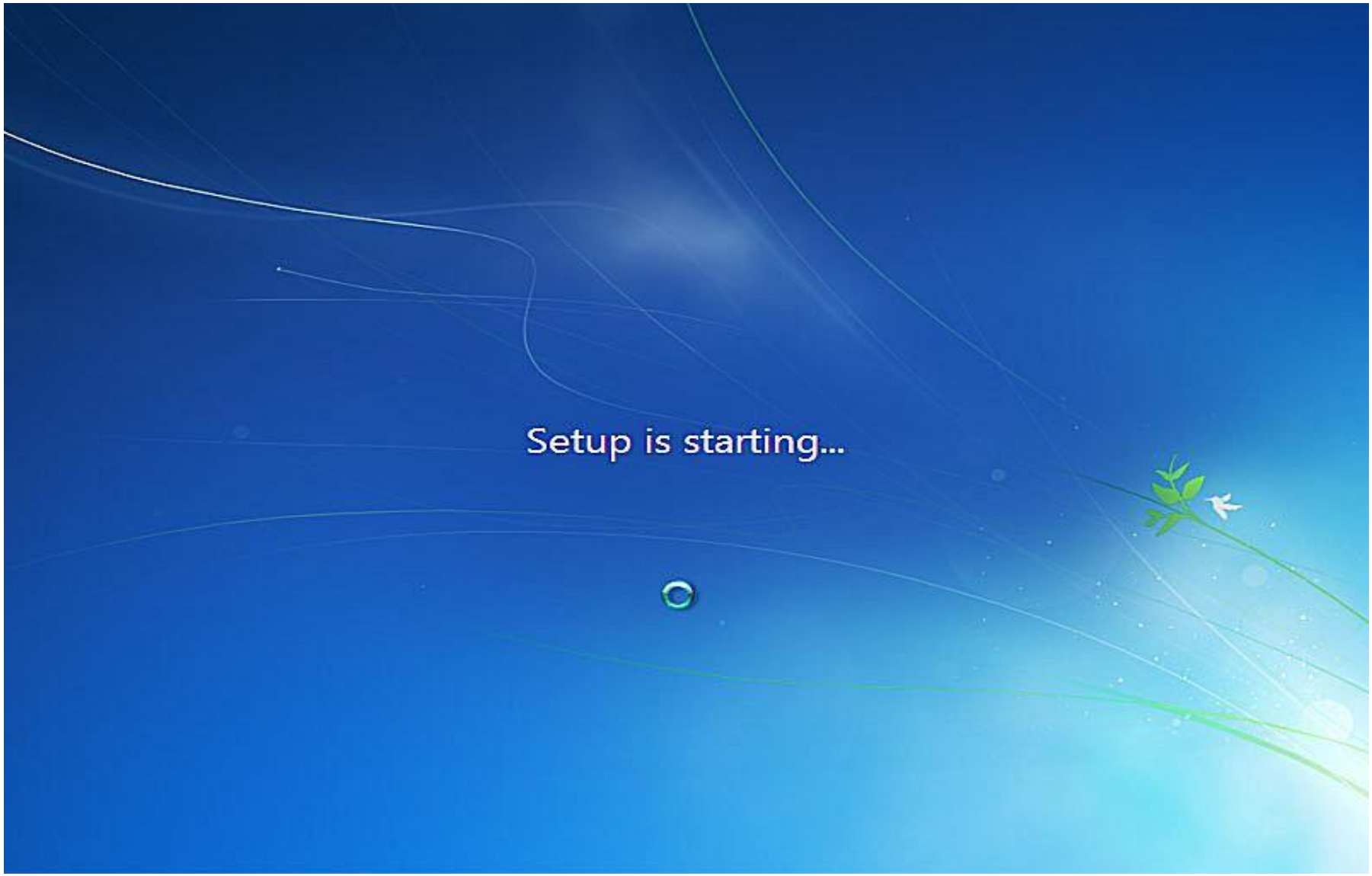
Step 4: Choose Language and Other Preferences



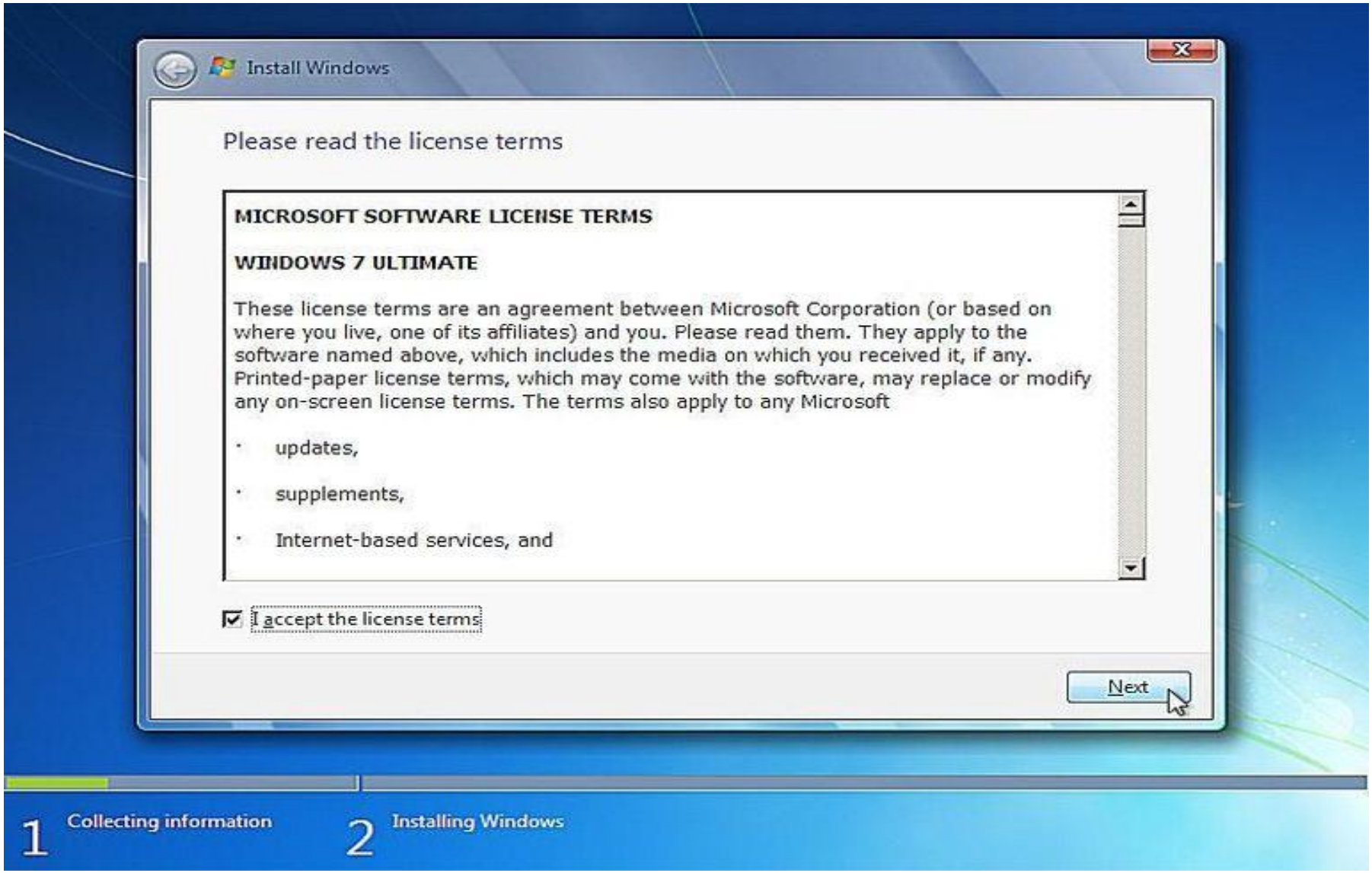
Step 5: Select the Install Now Button



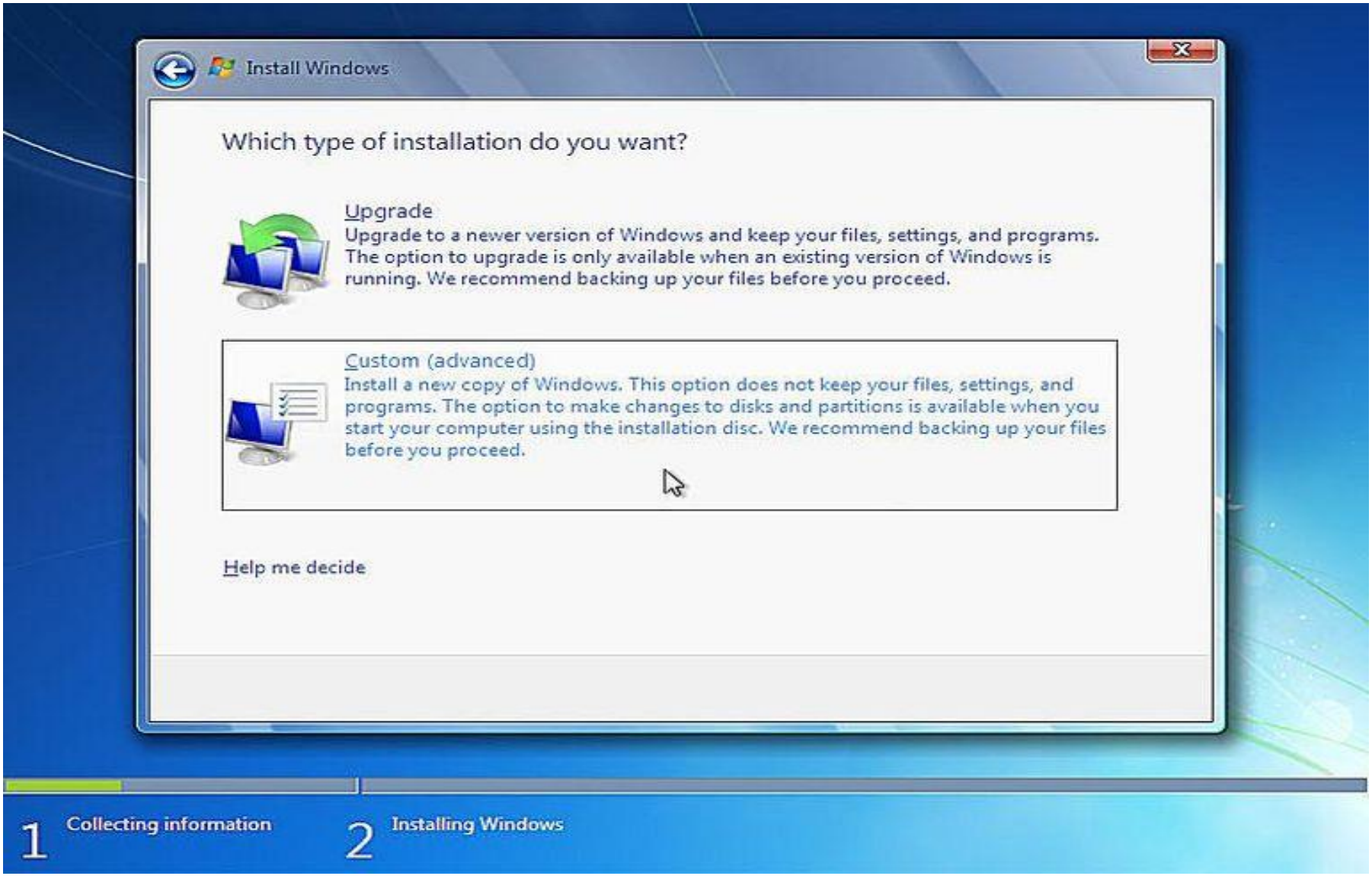
Step 6: Wait for Windows 7 Setup to Begin



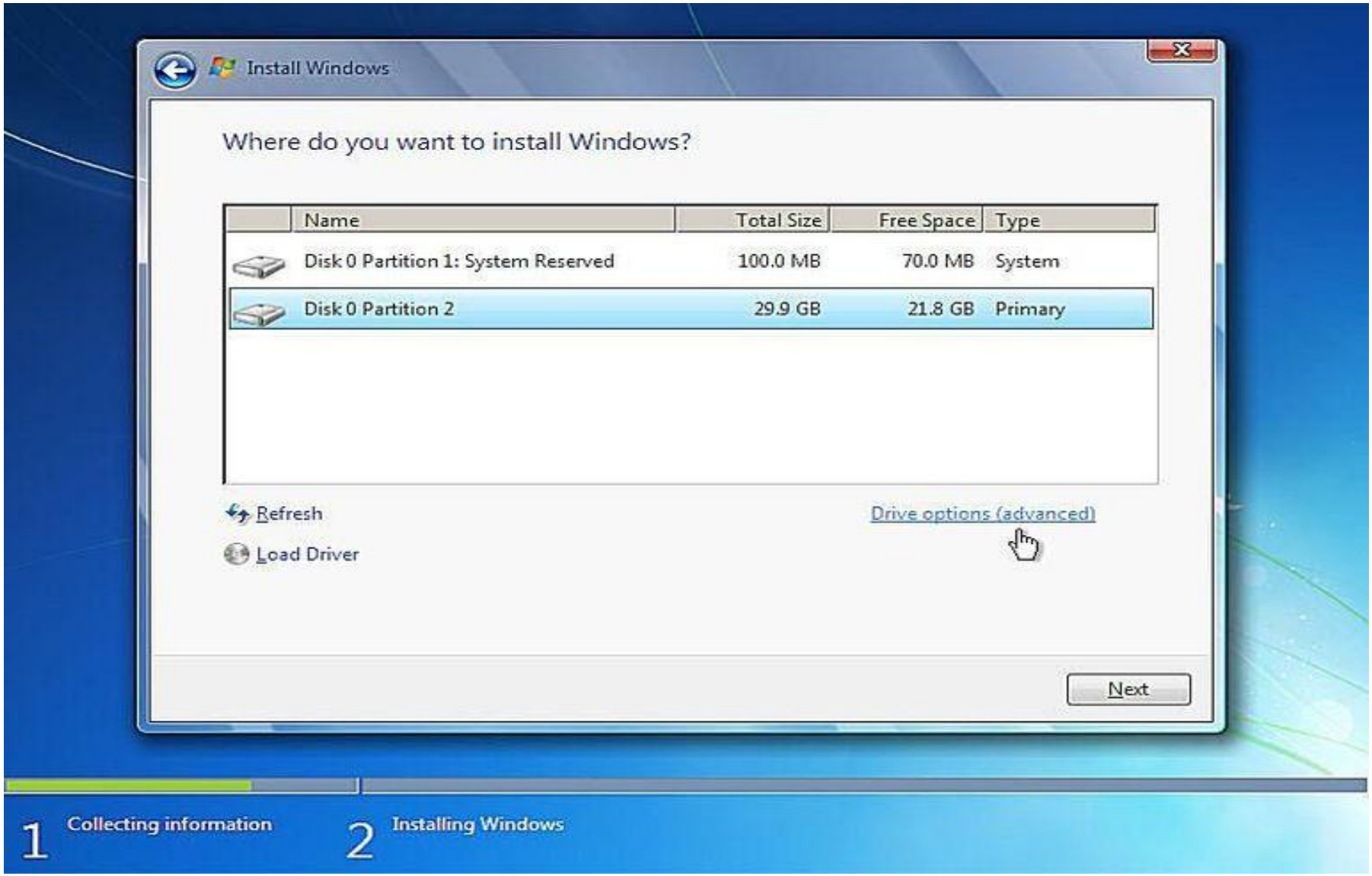
Step 7: Accept the Windows 7 License Terms



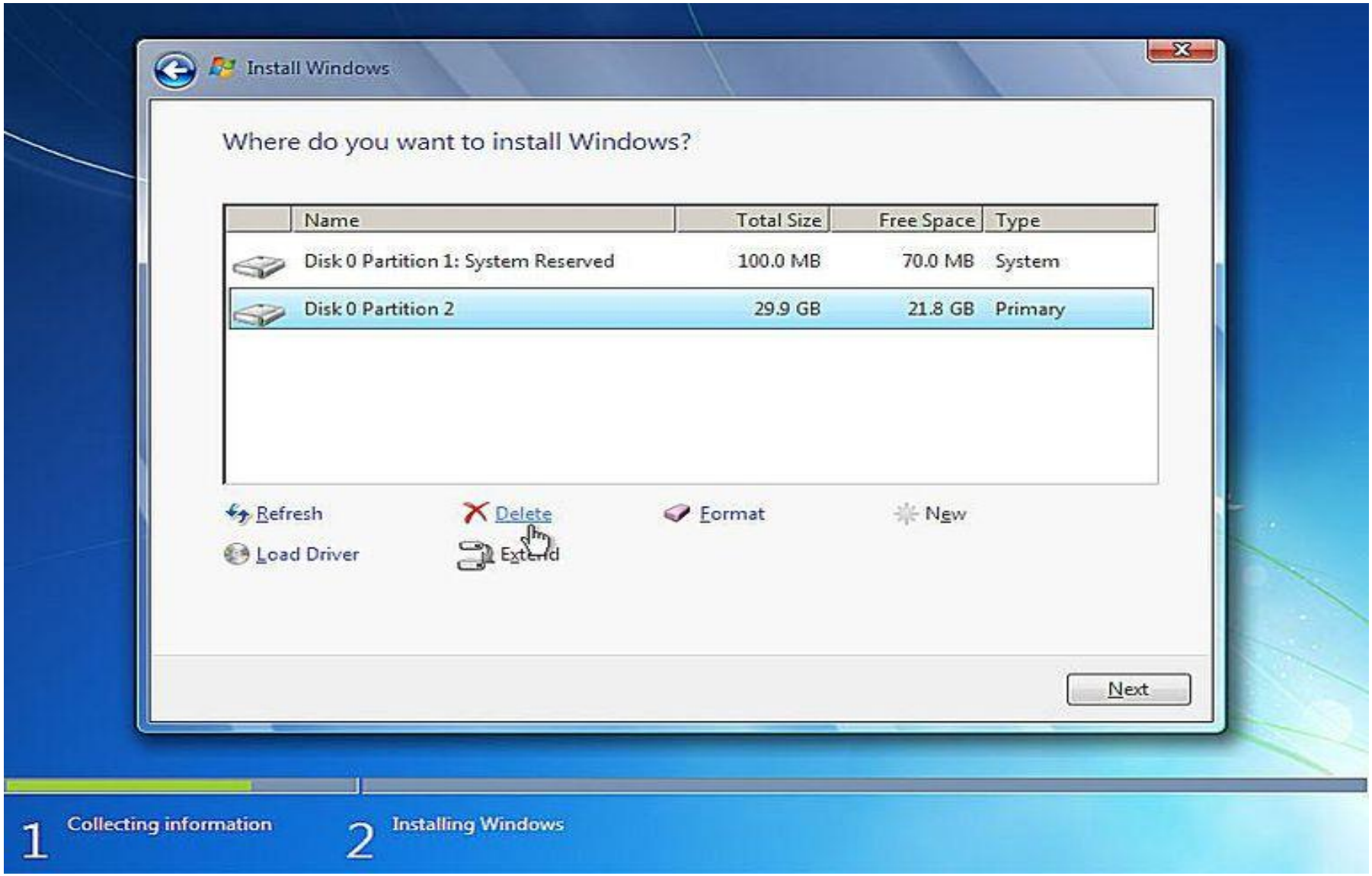
Step 8: Click on the Custom (advanced) button.



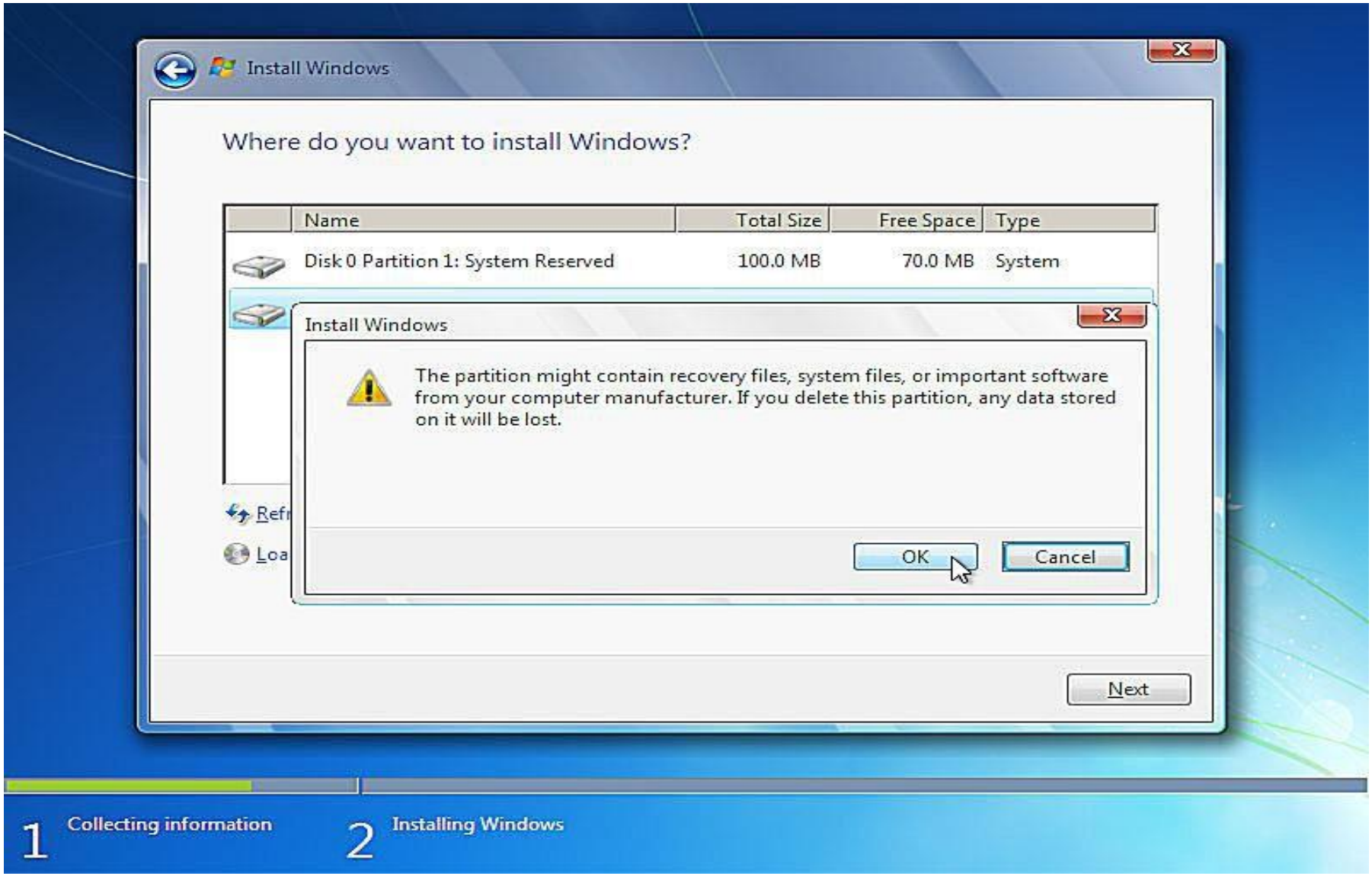
Step 9: Show the Windows 7 Advanced Drive Options



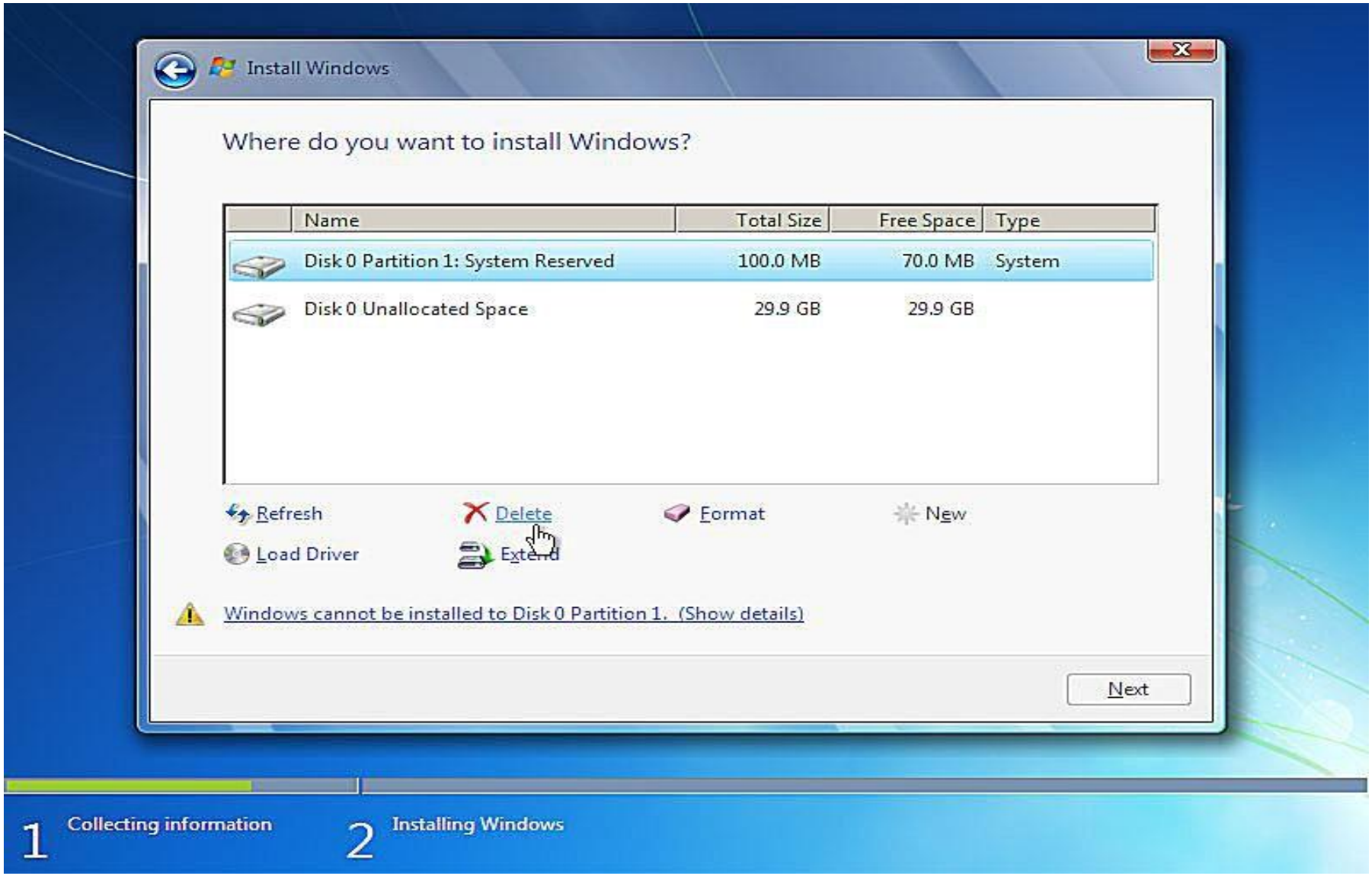
Step 10: Delete the Partition Windows is Installed On



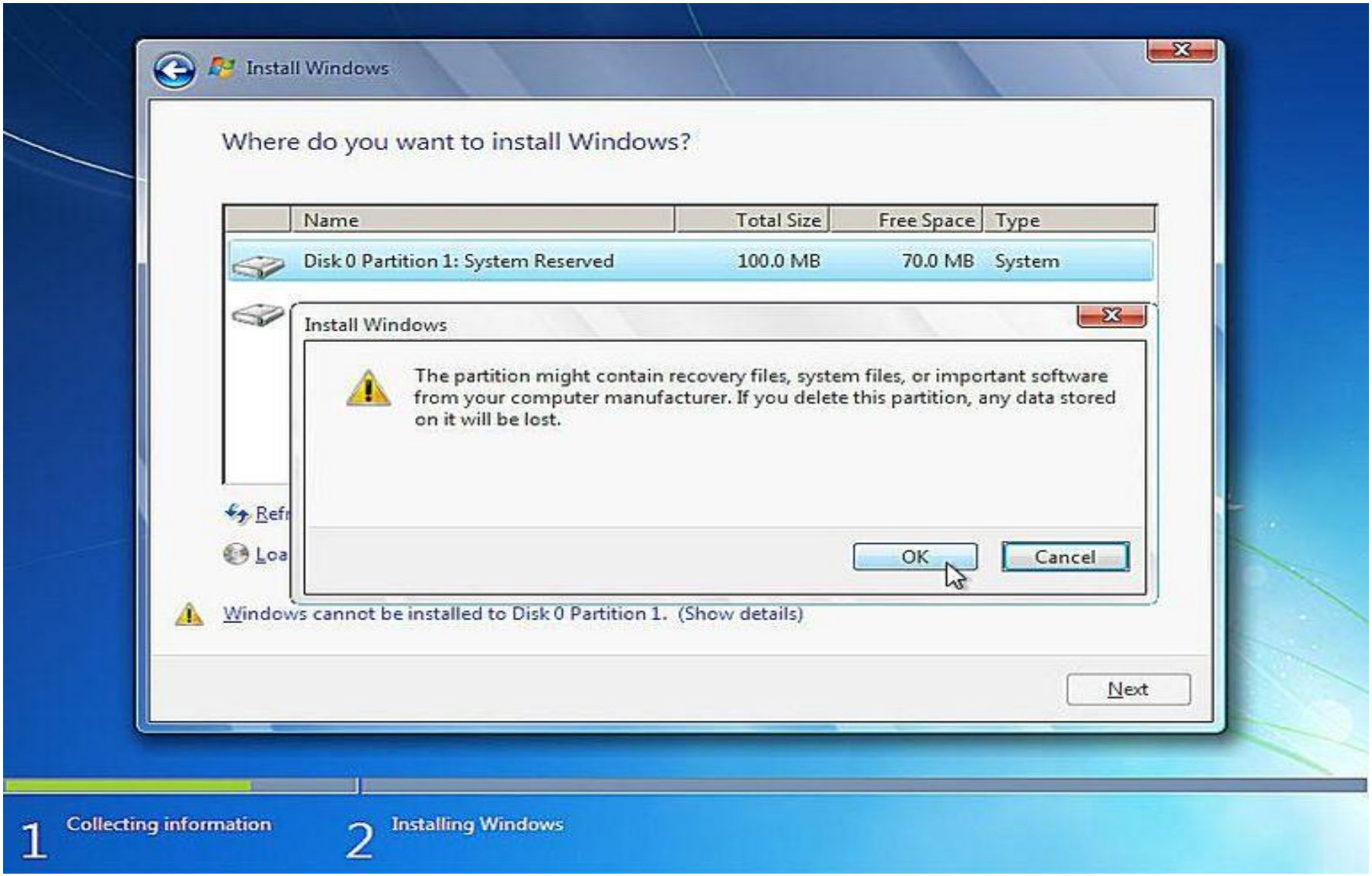
Step 11: Confirm the Partition Deletion



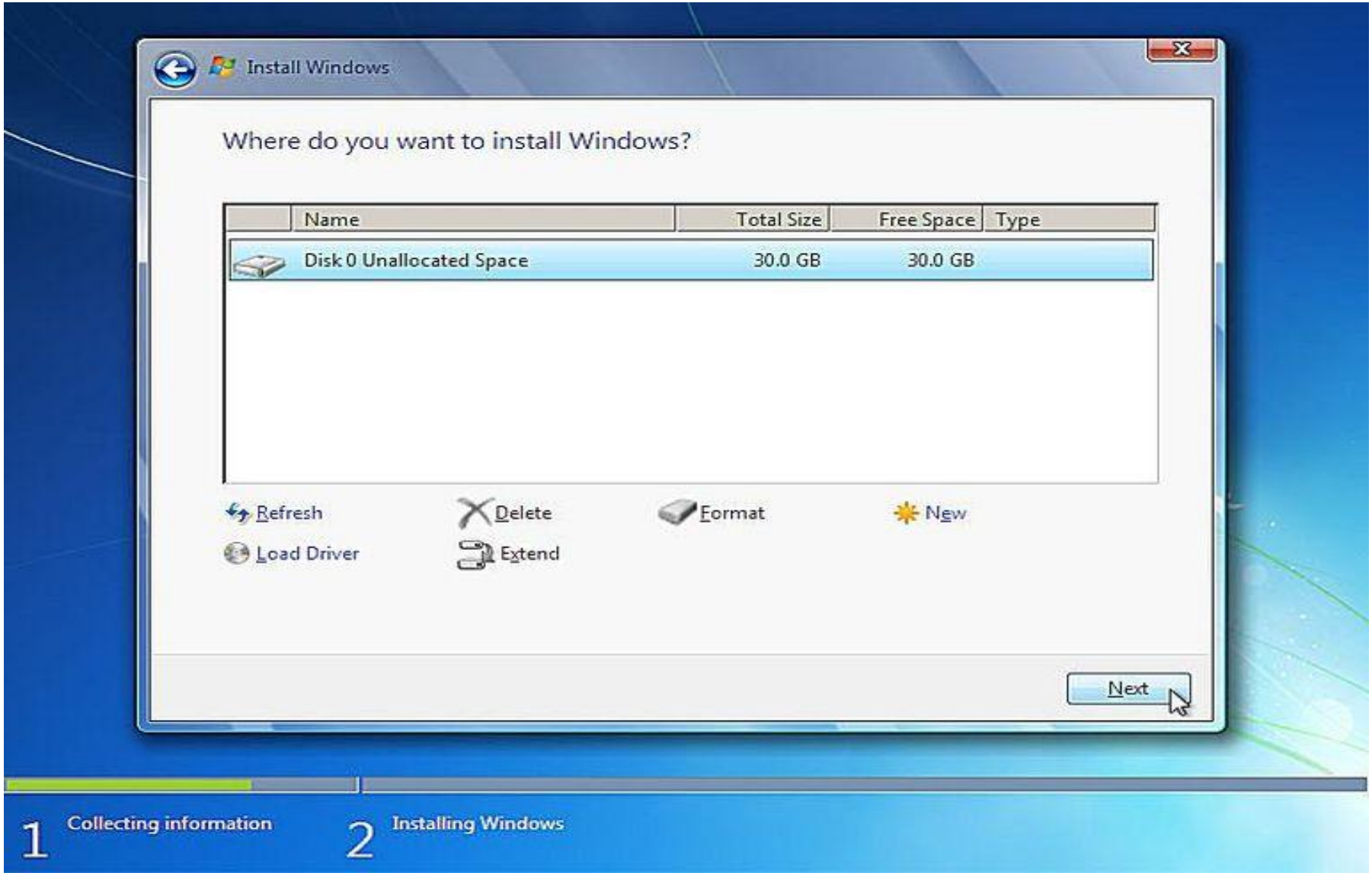
Step 12: Delete Other Operating System Related Partitions



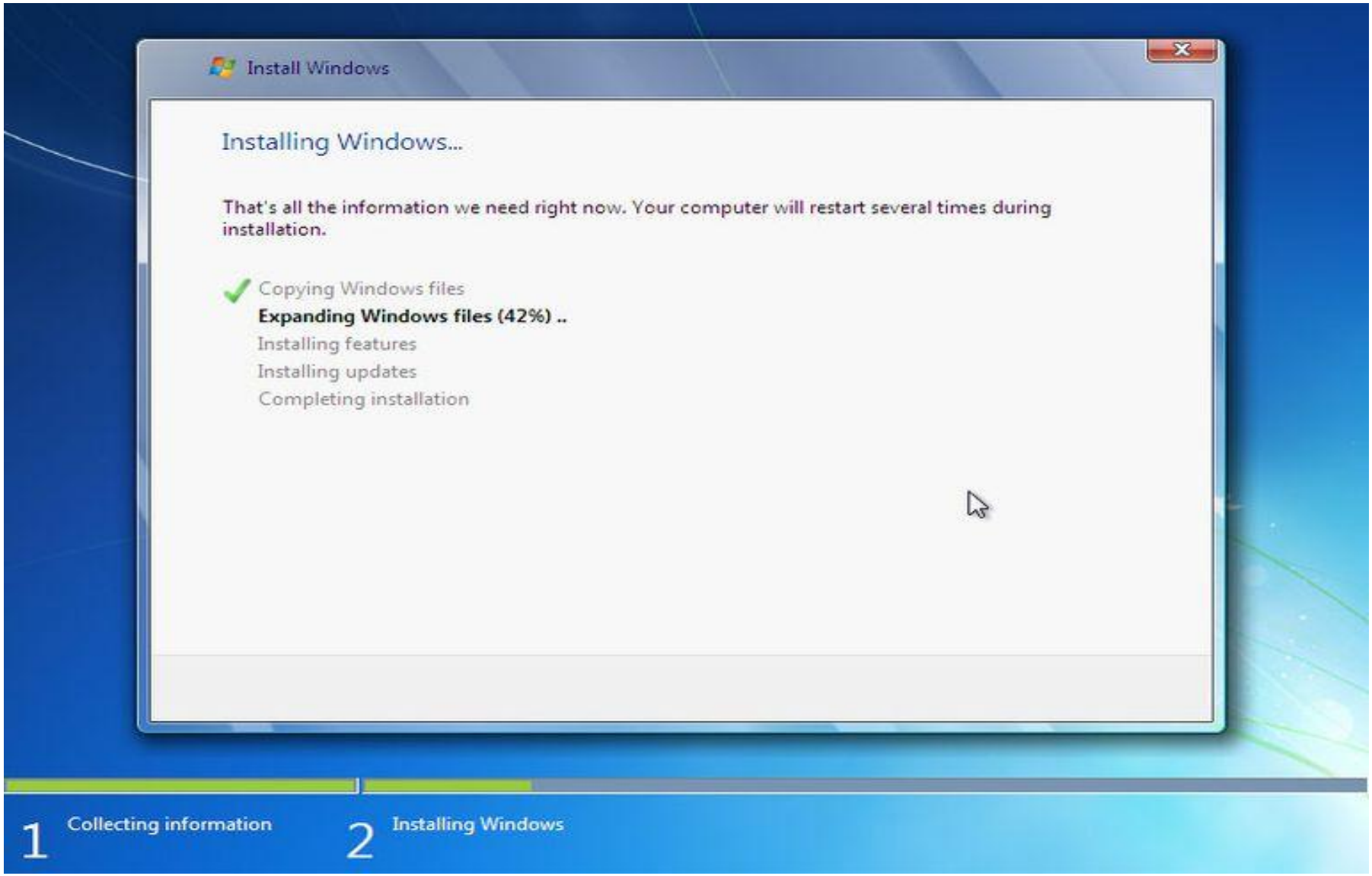
Step 13: Confirm Delete Other Operating System Related Partitions



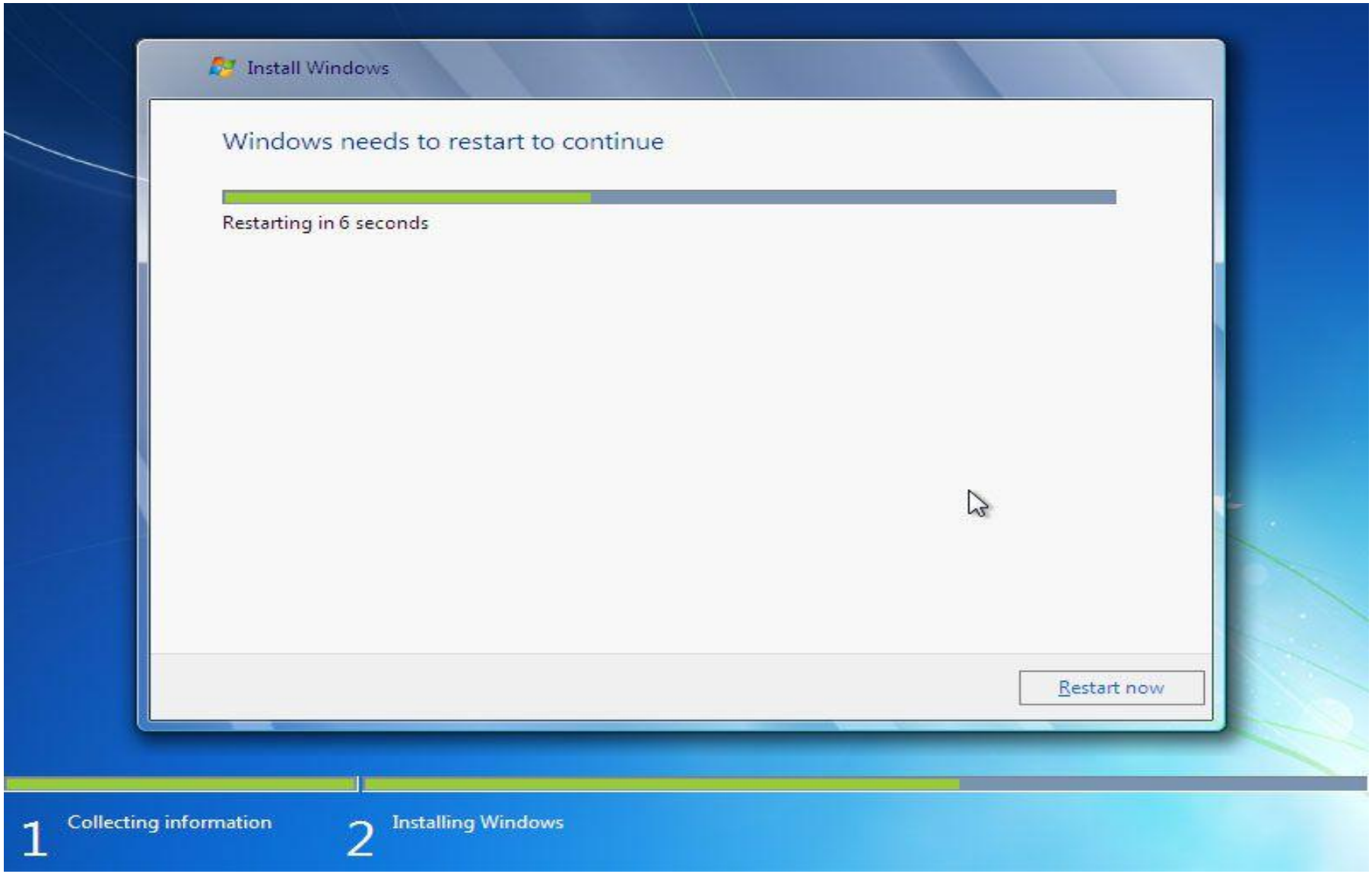
Step 14: Choose a Physical Location to Install Windows 7 On



Step 15: Wait While Windows 7 is Installed (the machine will restart several times)



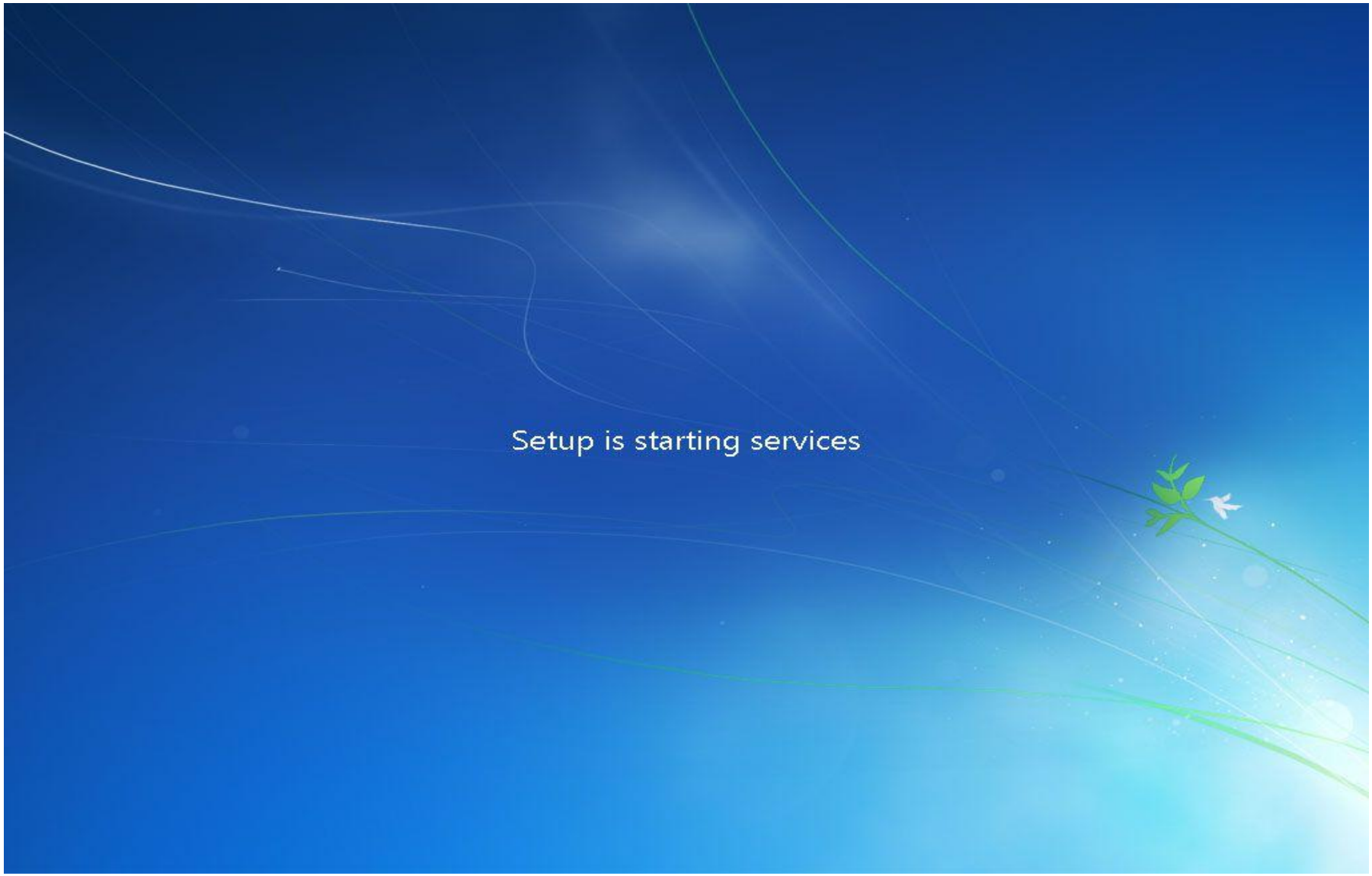
Step 16: After Completing the Installation Restart the System (First Time)



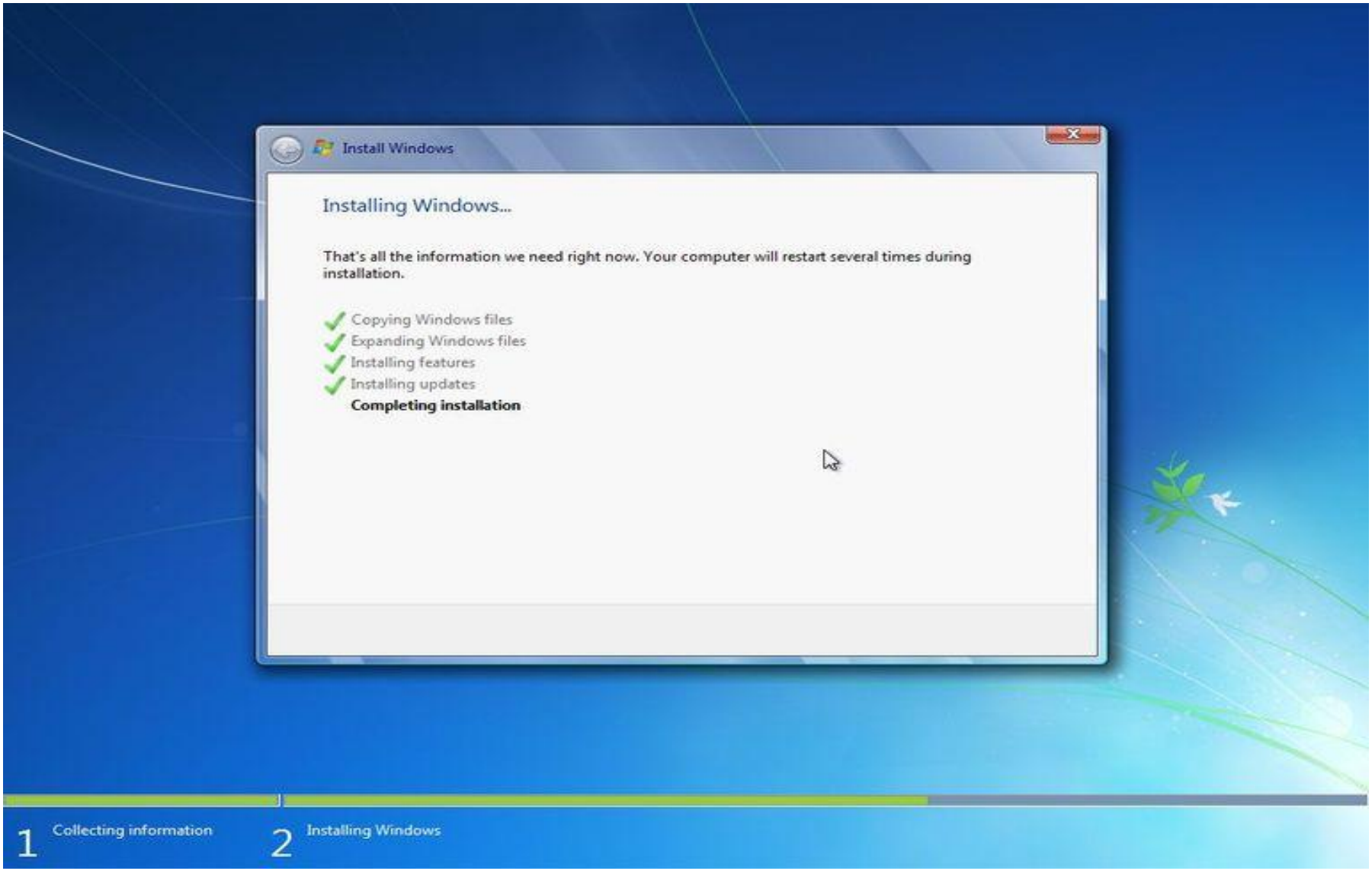
Step 17: Wait for Windows 7 Setup to Begin Again

Step 18: Wait for Windows 7 Setup to Update Registry Settings

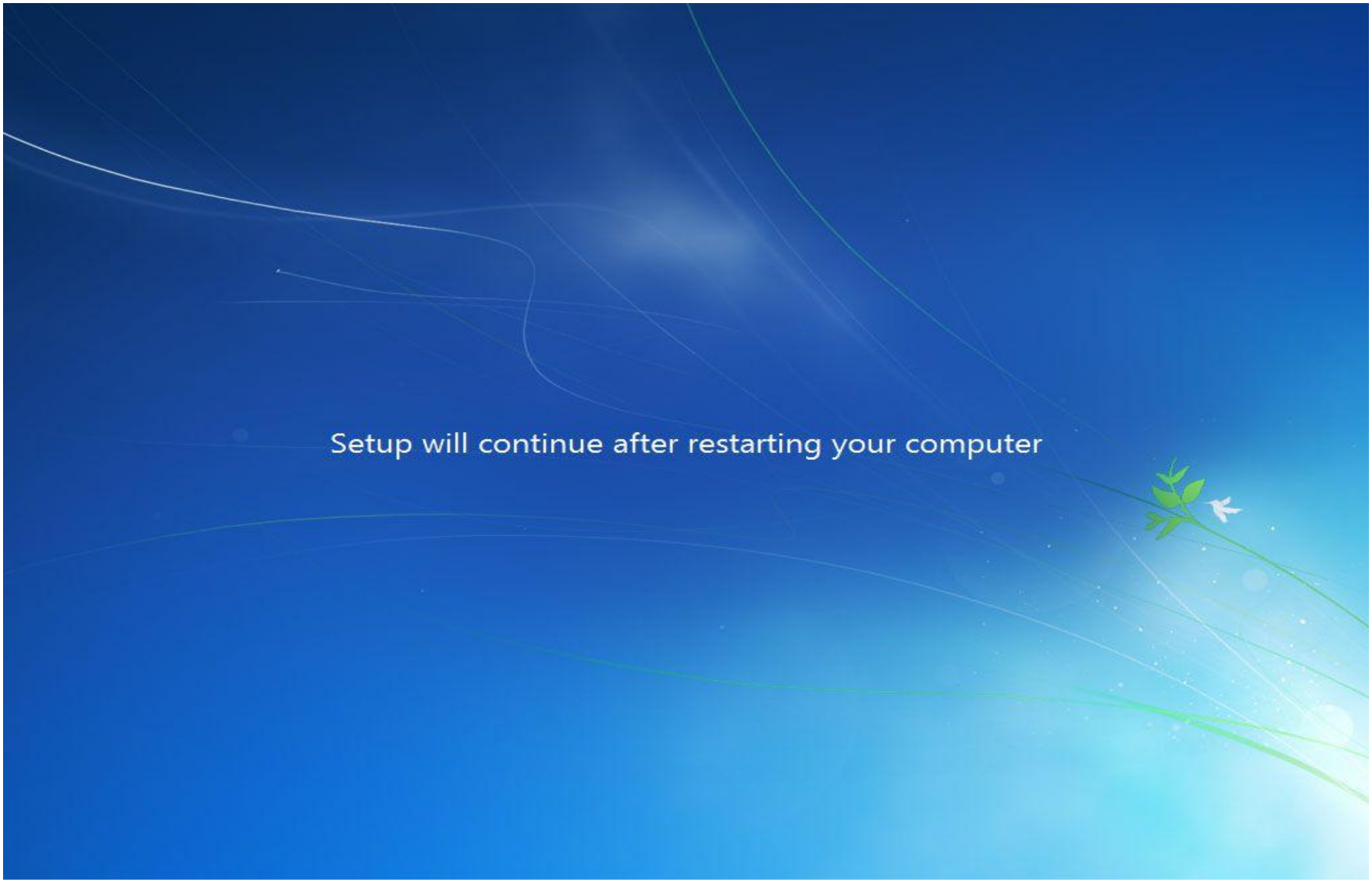
Step 19:Wait for Windows 7 Setup to Start Services



Step 20: Wait for Windows 7 Setup to Complete

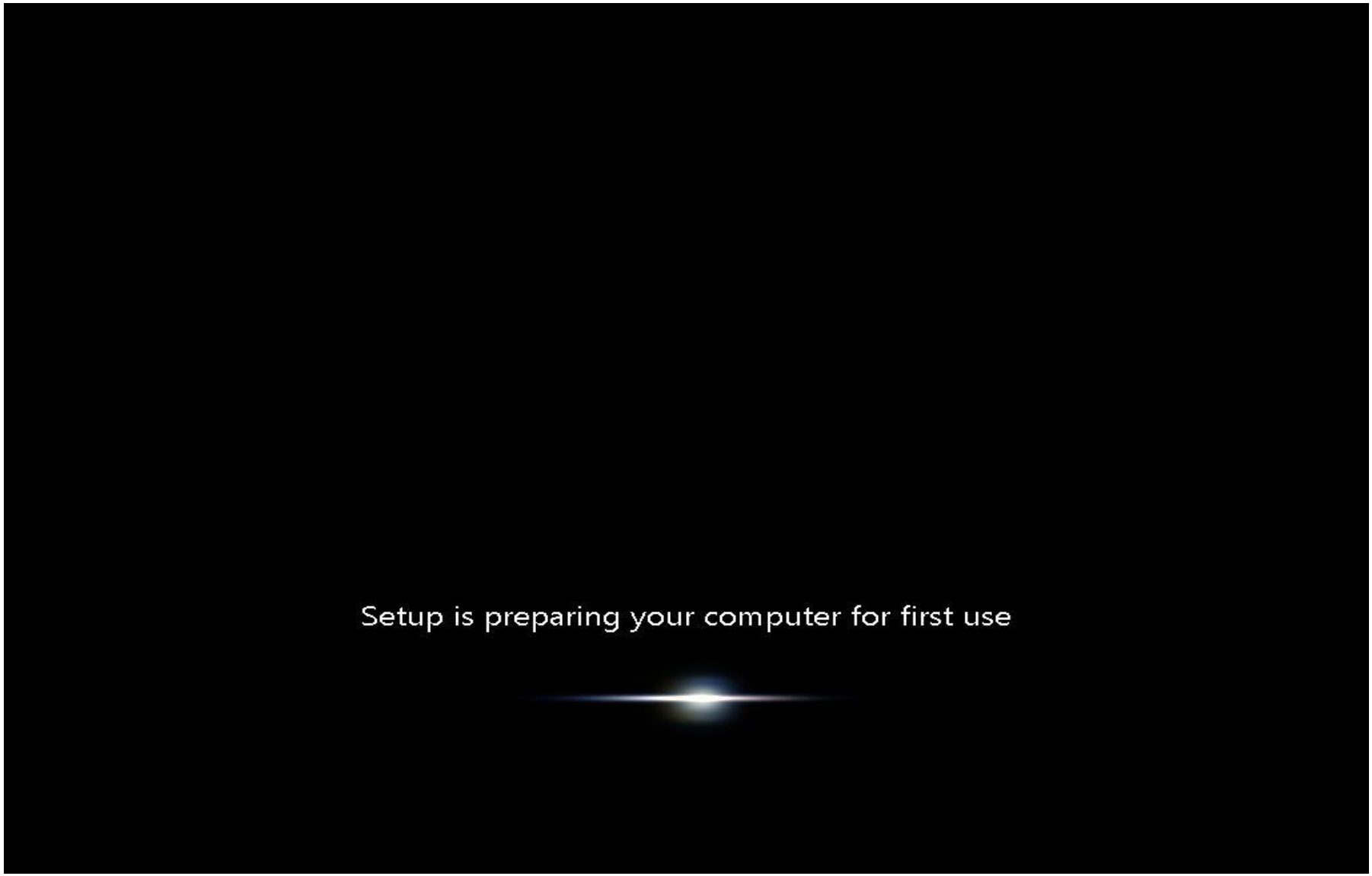


Step 21:Wait for Your PC to Automatically Restart



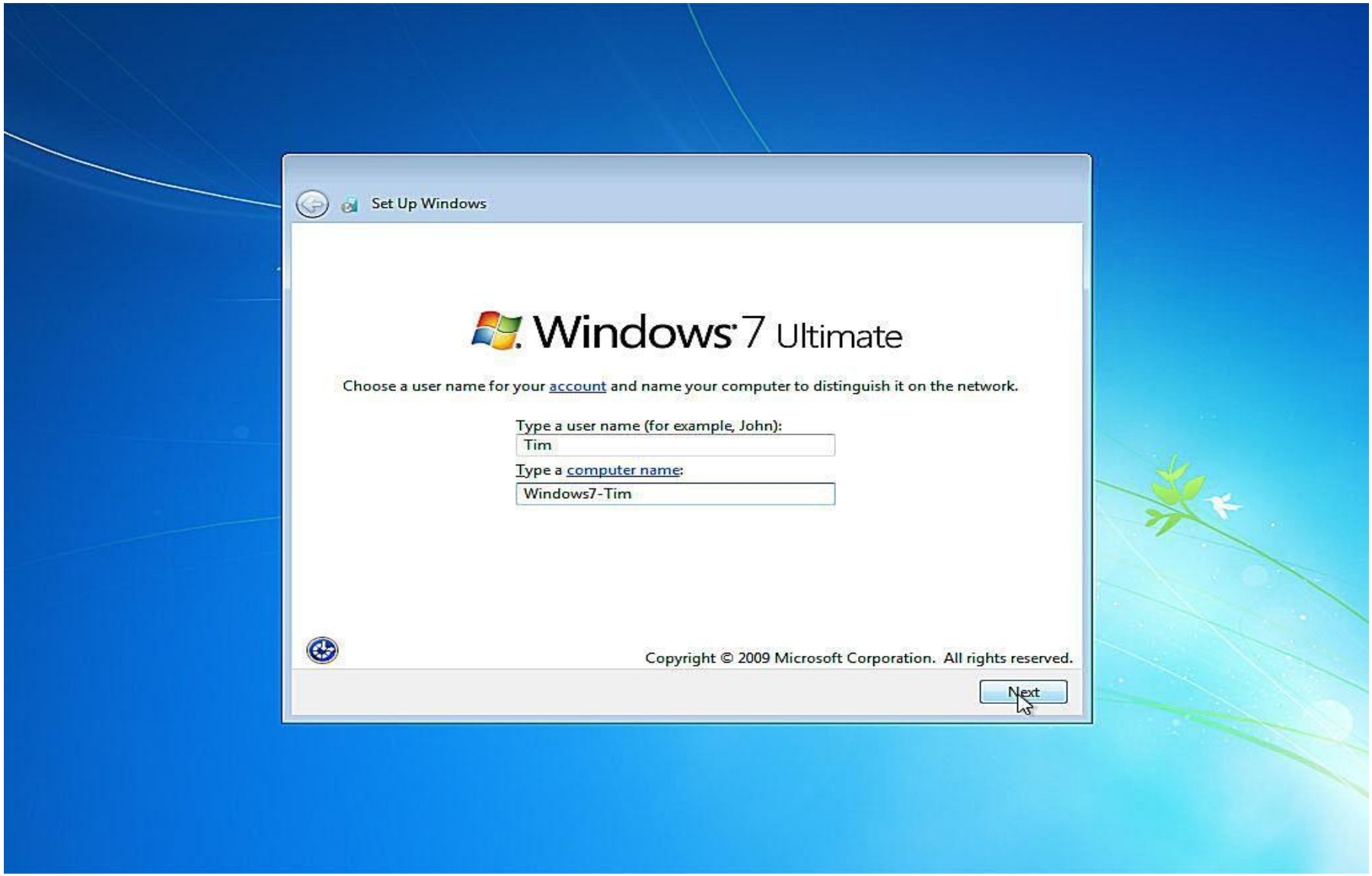
Step 22:Wait for Windows 7 to Start

Step 23: Wait for Windows 7 to Prepare Your PC for First Use

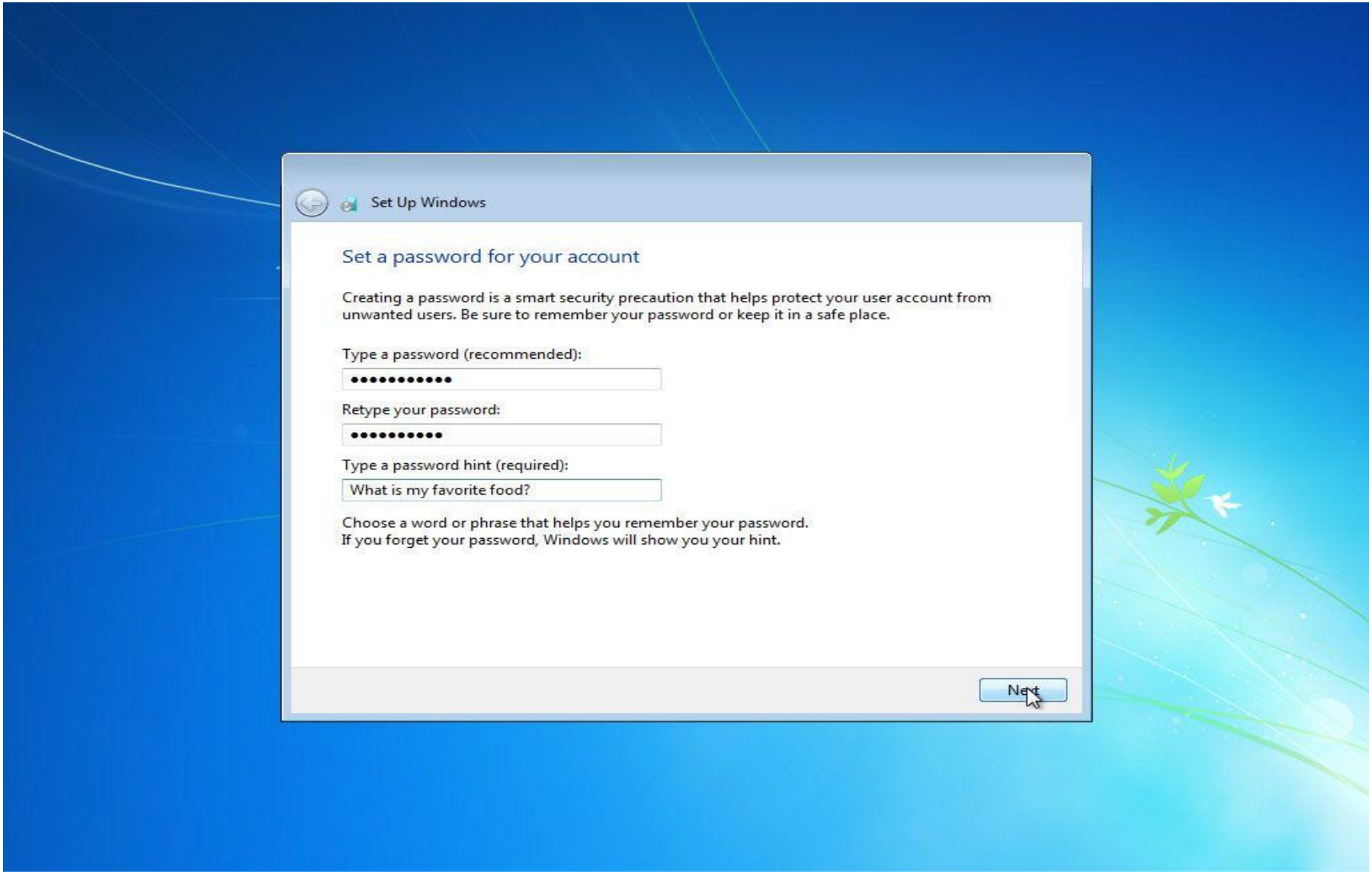


Step 24: Wait for Windows 7 to Check Your PC's Video Performance

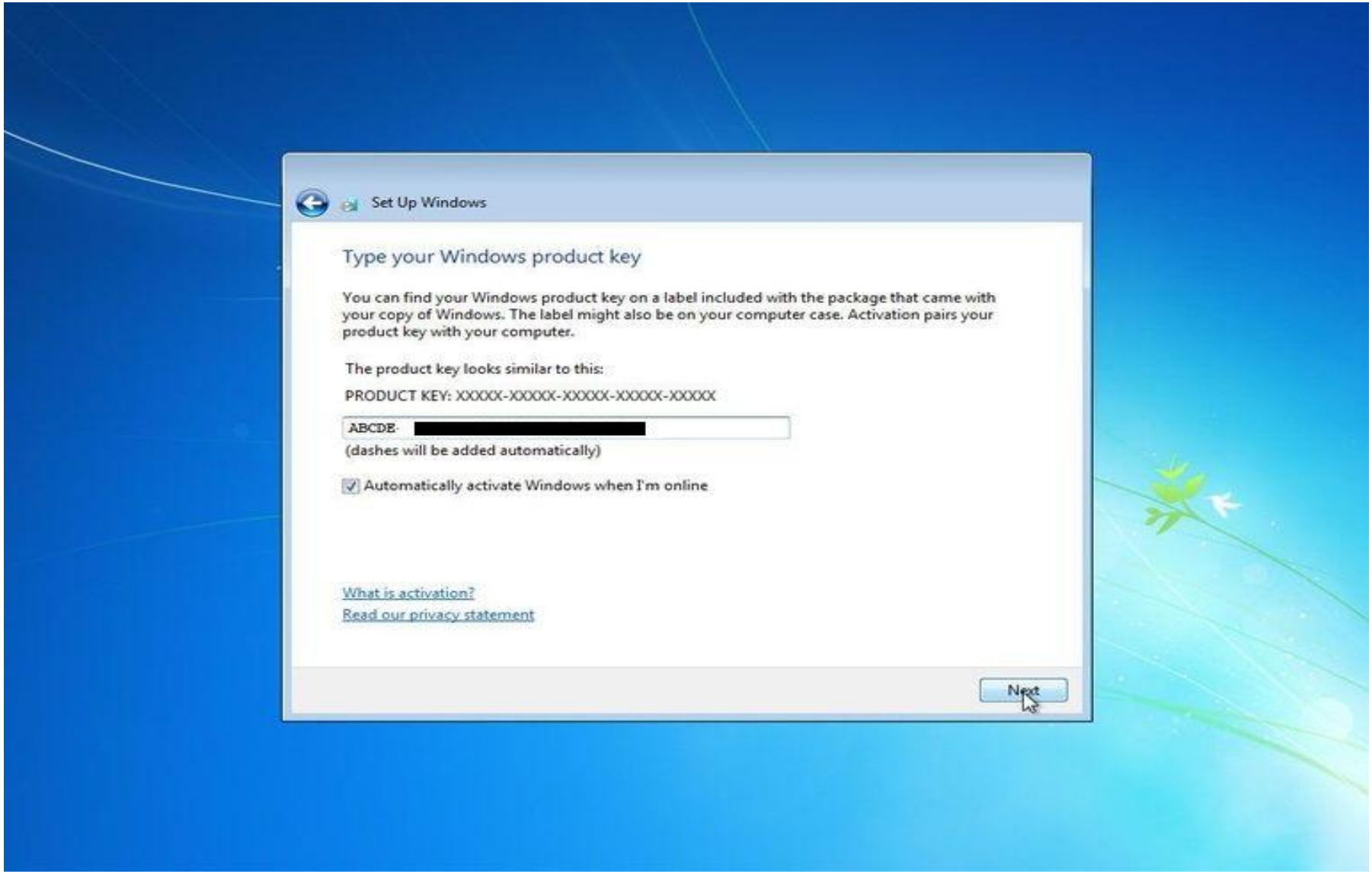
Step 25: After Completing the Installation Enter Username and Computer Name



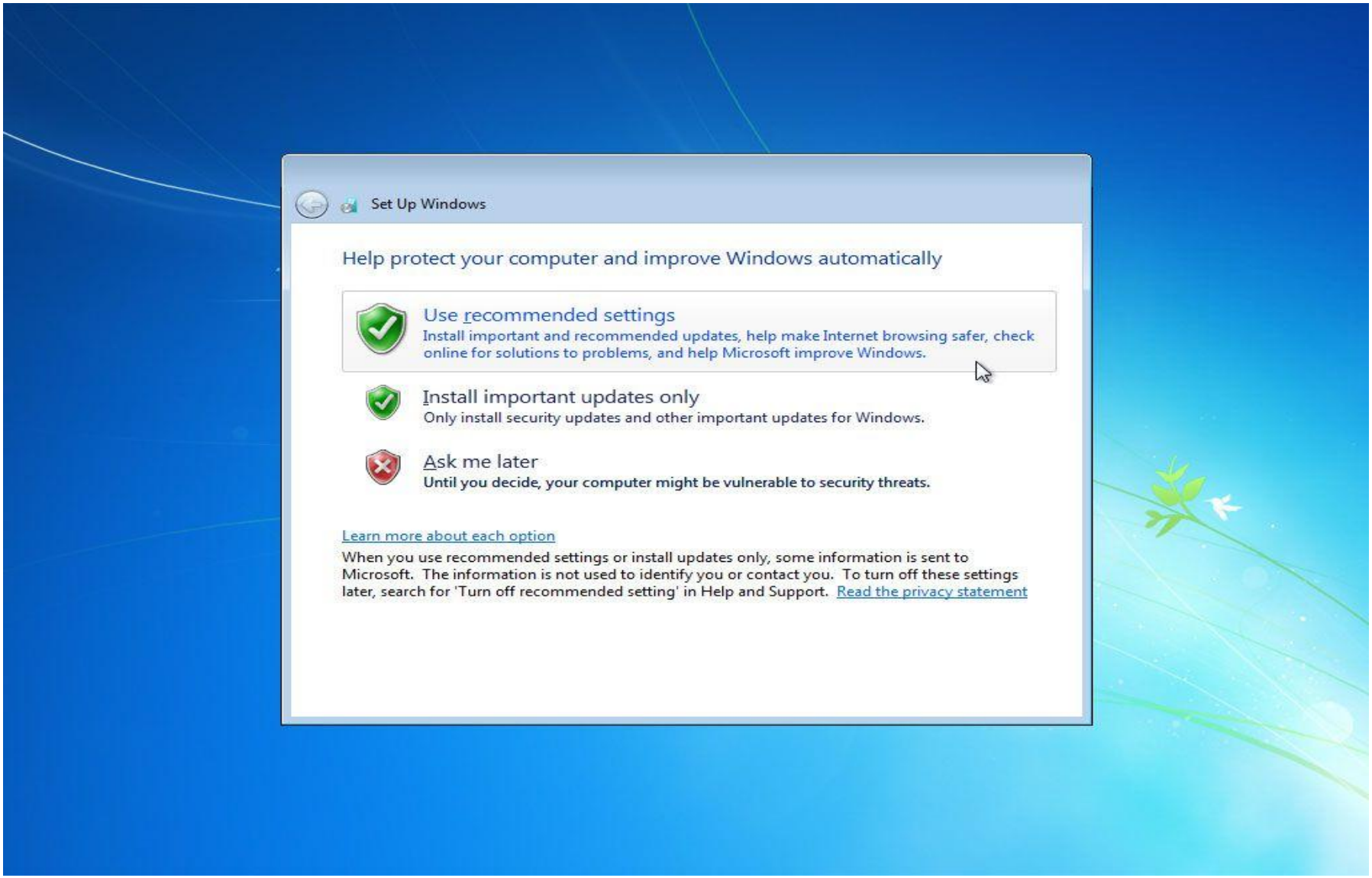
Step 26: Choose a Password to Access Windows 7



Step 27: Enter the Windows 7 Product Key

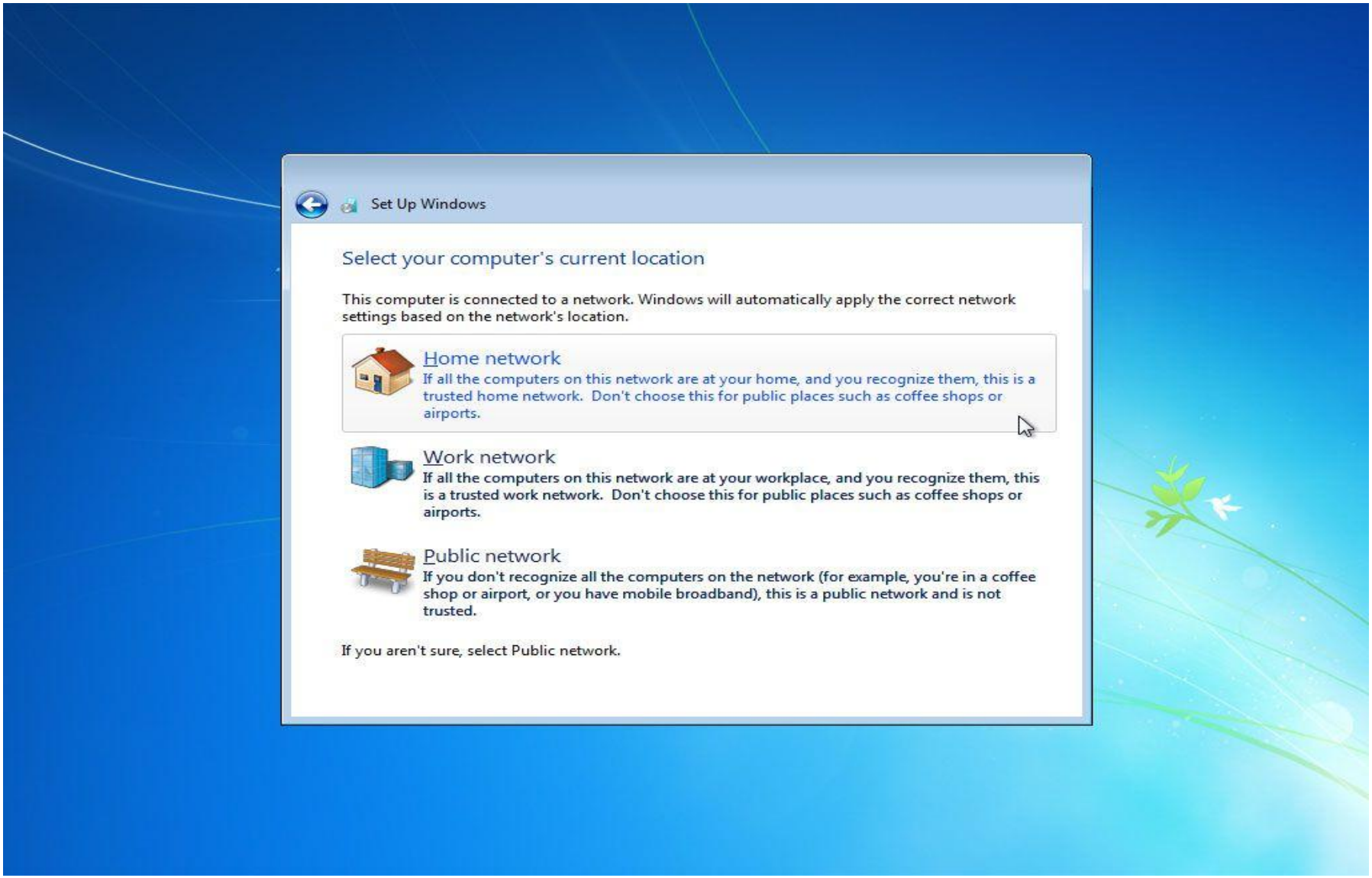


Step 28:Choose a Windows Update Option

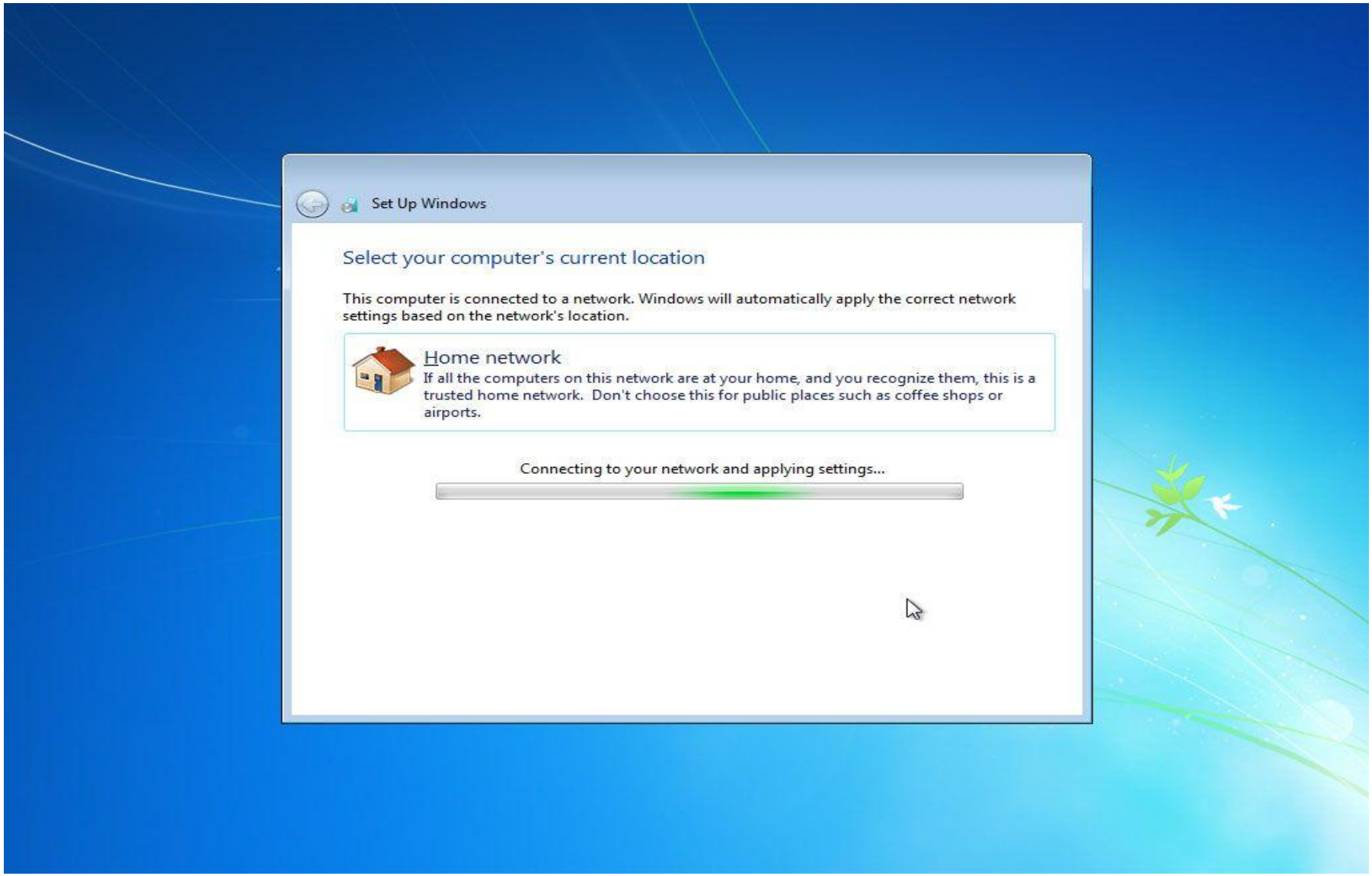


Step 29: Choose the Correct Time Zone, Date, and Time

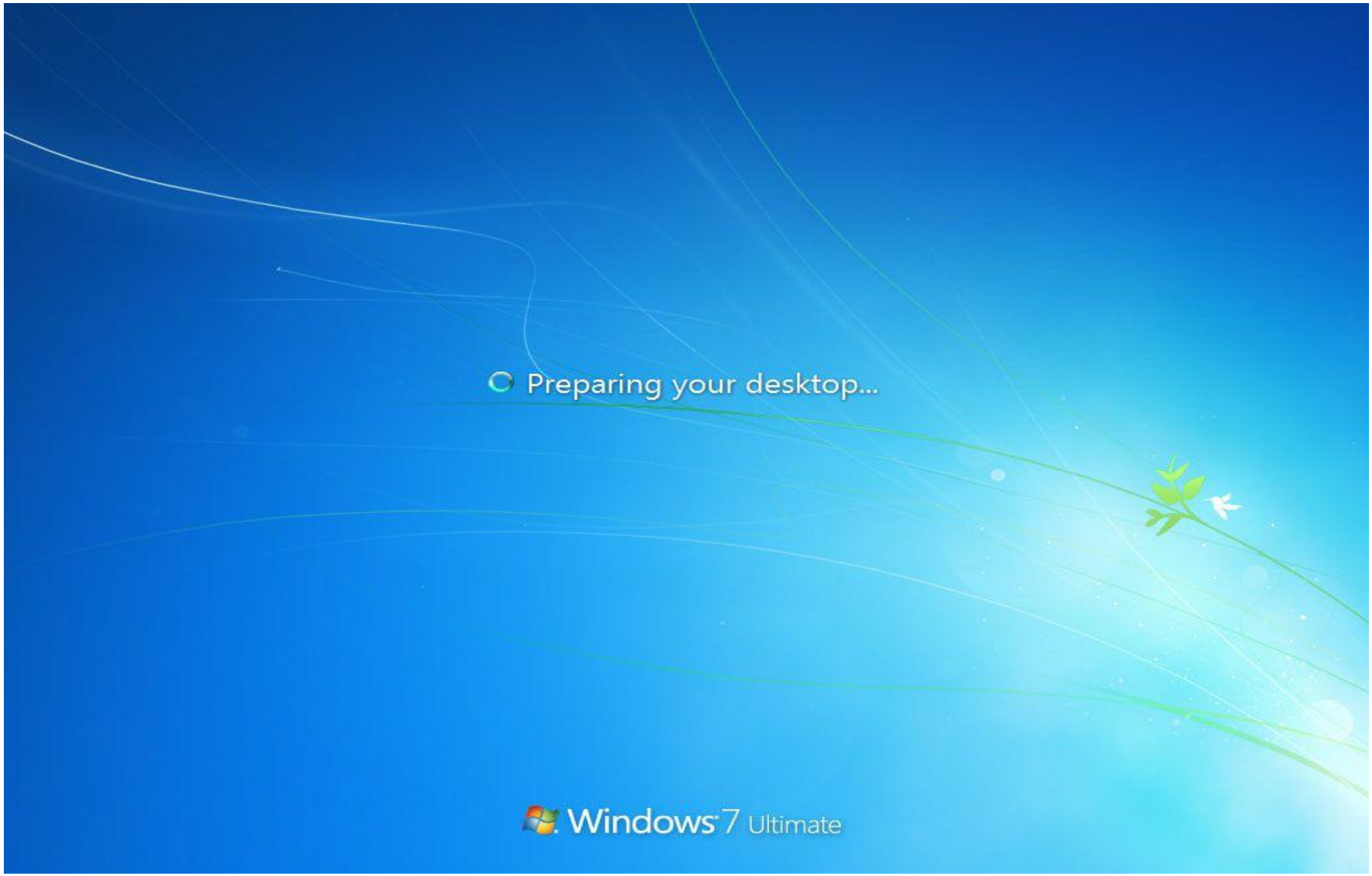
Step 30: Choose a Network Location



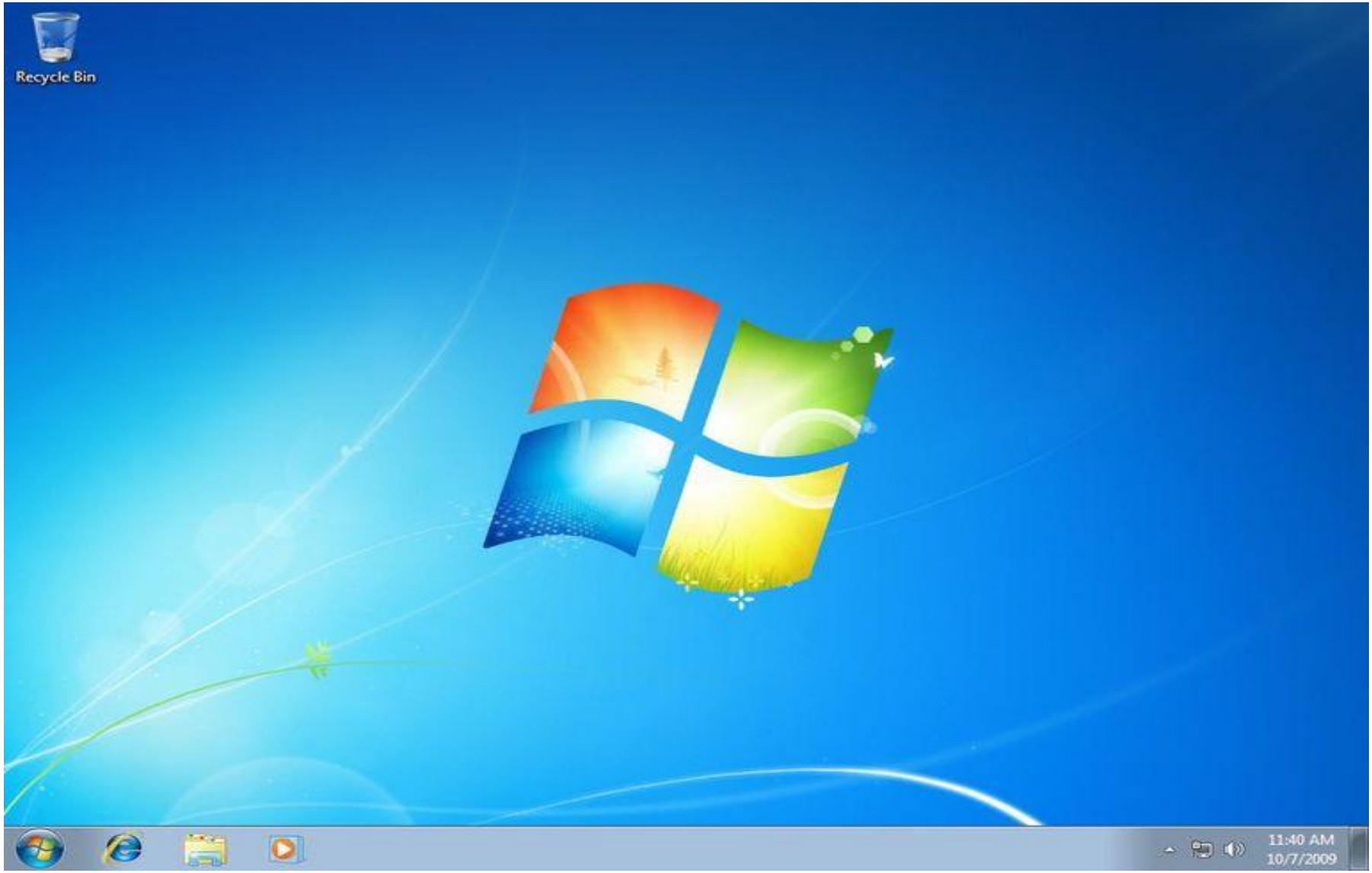
Step 31: Wait for Windows 7 to Connect to the Network



Step 32: Wait for Windows 7 to Prepare the Desktop



Step 33: Your Windows 7 Clean Install is Complete!



OS Installation Ubuntu 16.04LTS

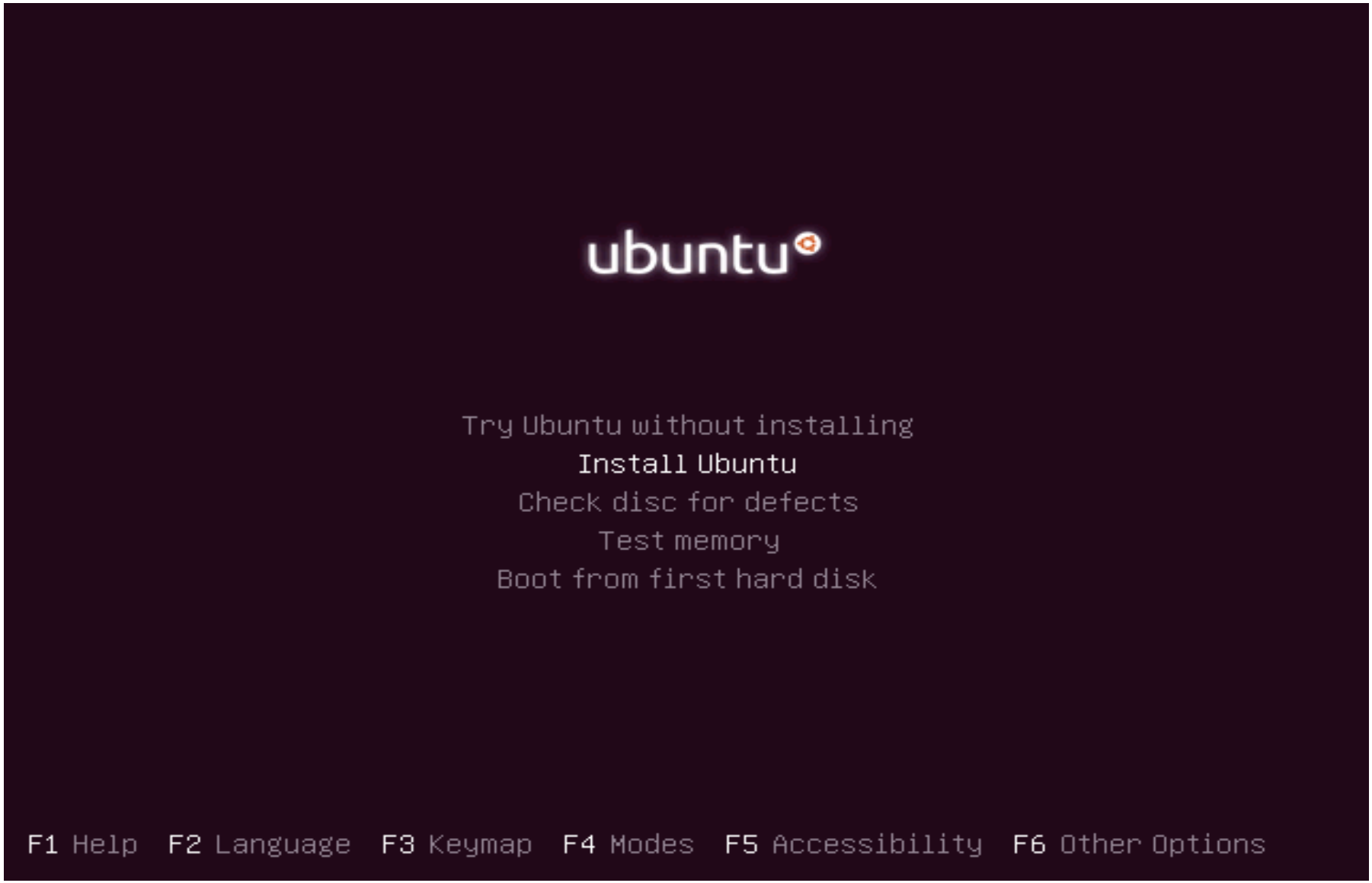
Requirements

You'll need to consider the following before starting the installation:

- ❑ Connect your **Desktop/Laptop** to a power source.
- ❑ Ensure you have at least 5GB of free storage space.
- ❑ Have access to either a DVD or a USB flash drive containing the version of Ubuntu you want to install.
- ❑ Make sure you have a recent backup of your data. While it's unlikely that anything will go wrong, you can never be too prepared.
- ❑ Put the CD into the CD-ROM drive, change the boot sequence so that CD-ROM can boot first.
- ❑ If you want to **install Ubuntu from the USB flash Disk,** change the boot sequence according to the USB mass storage to boot first

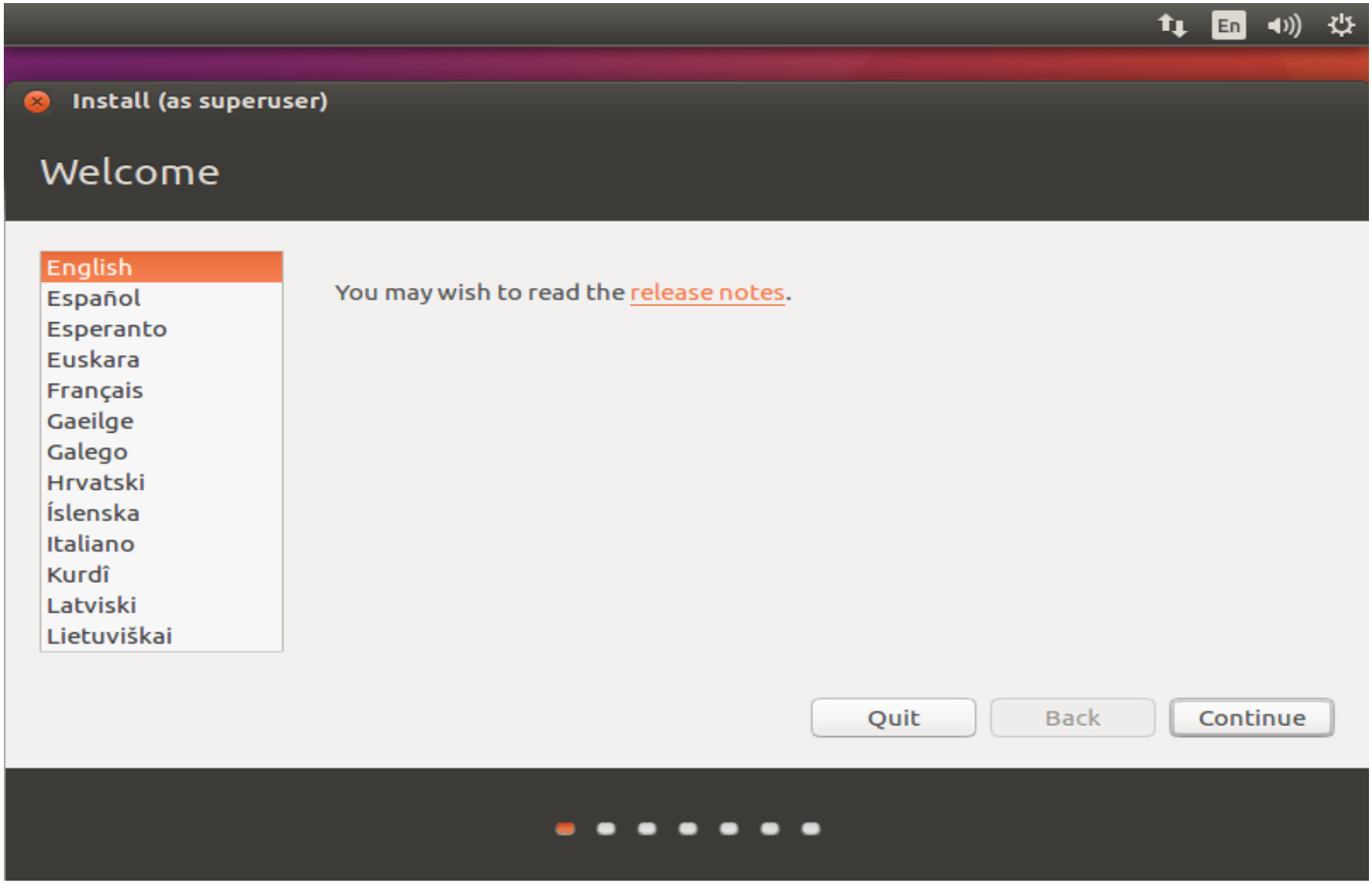
Press Enter to get a language screen and then select the language of your choice

For installing the Ubuntu 16.04, Select Install Ubuntu.

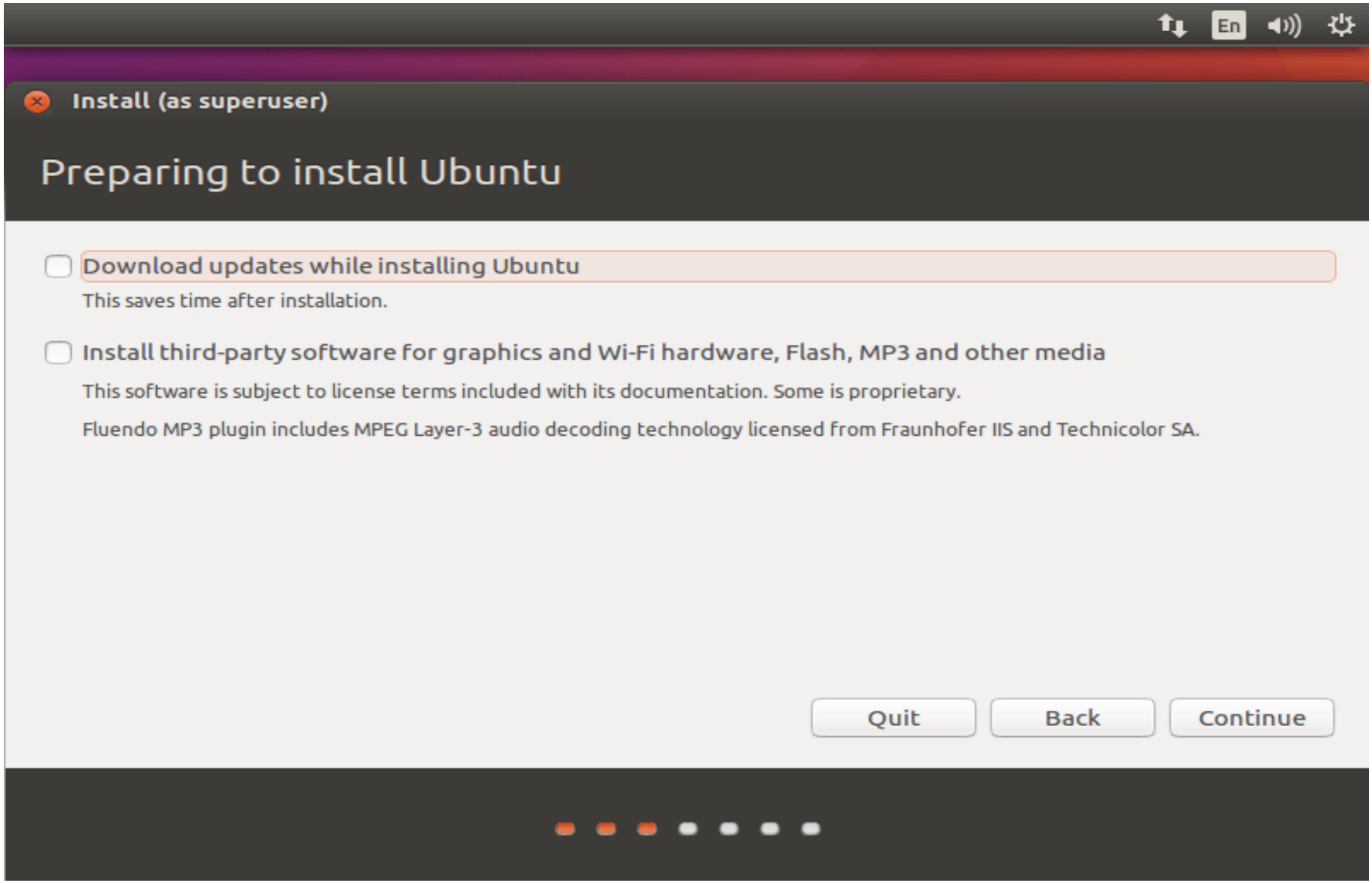


This is a starting screen, it will disappear in a minute

Click continue on the welcome screen.



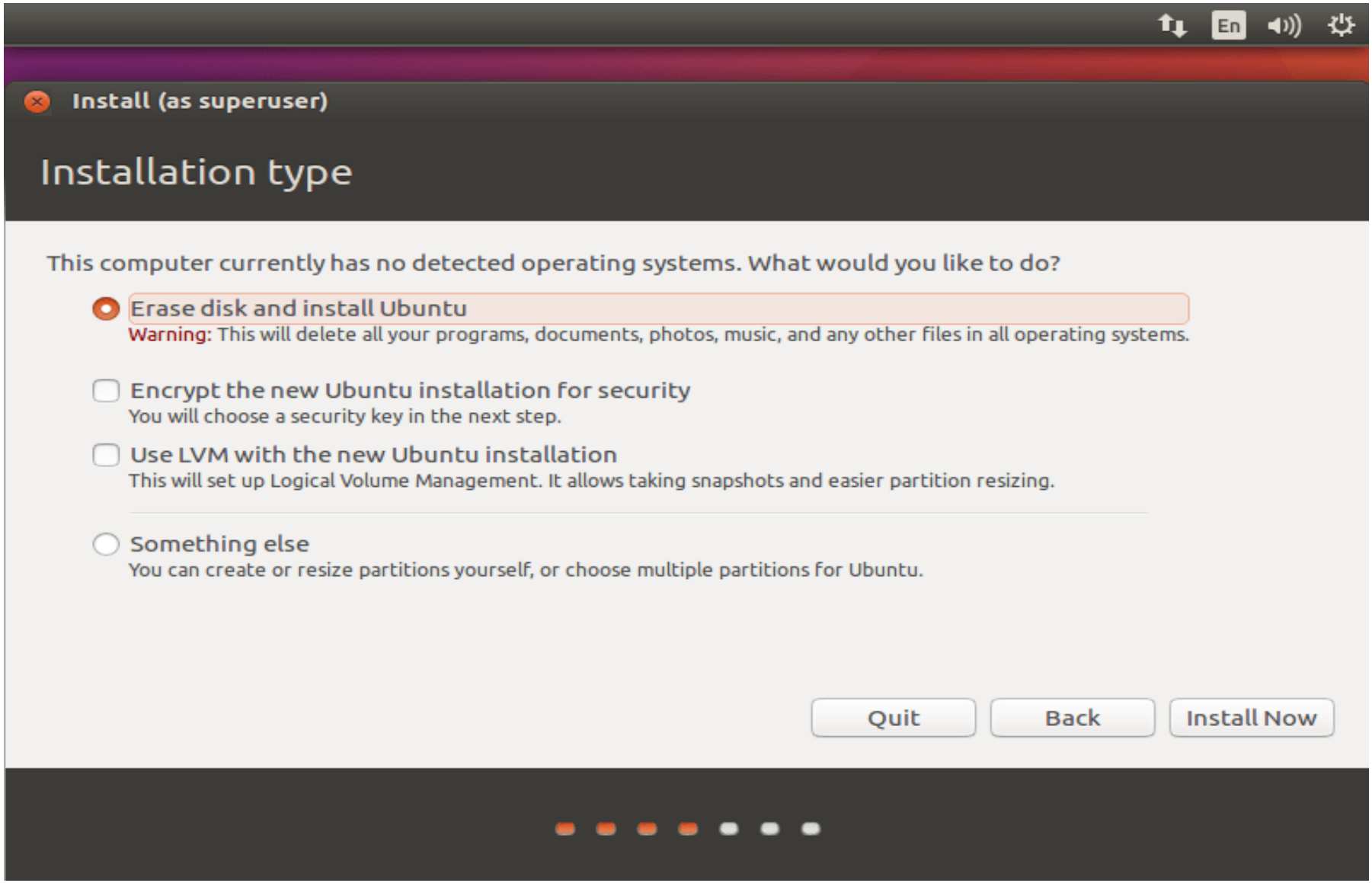
You can either choose to install updates and other third-party software while installing Ubuntu 16.04 or leave as it is since it requires internet and installation may take an hour-long depends on the downloadable contents.

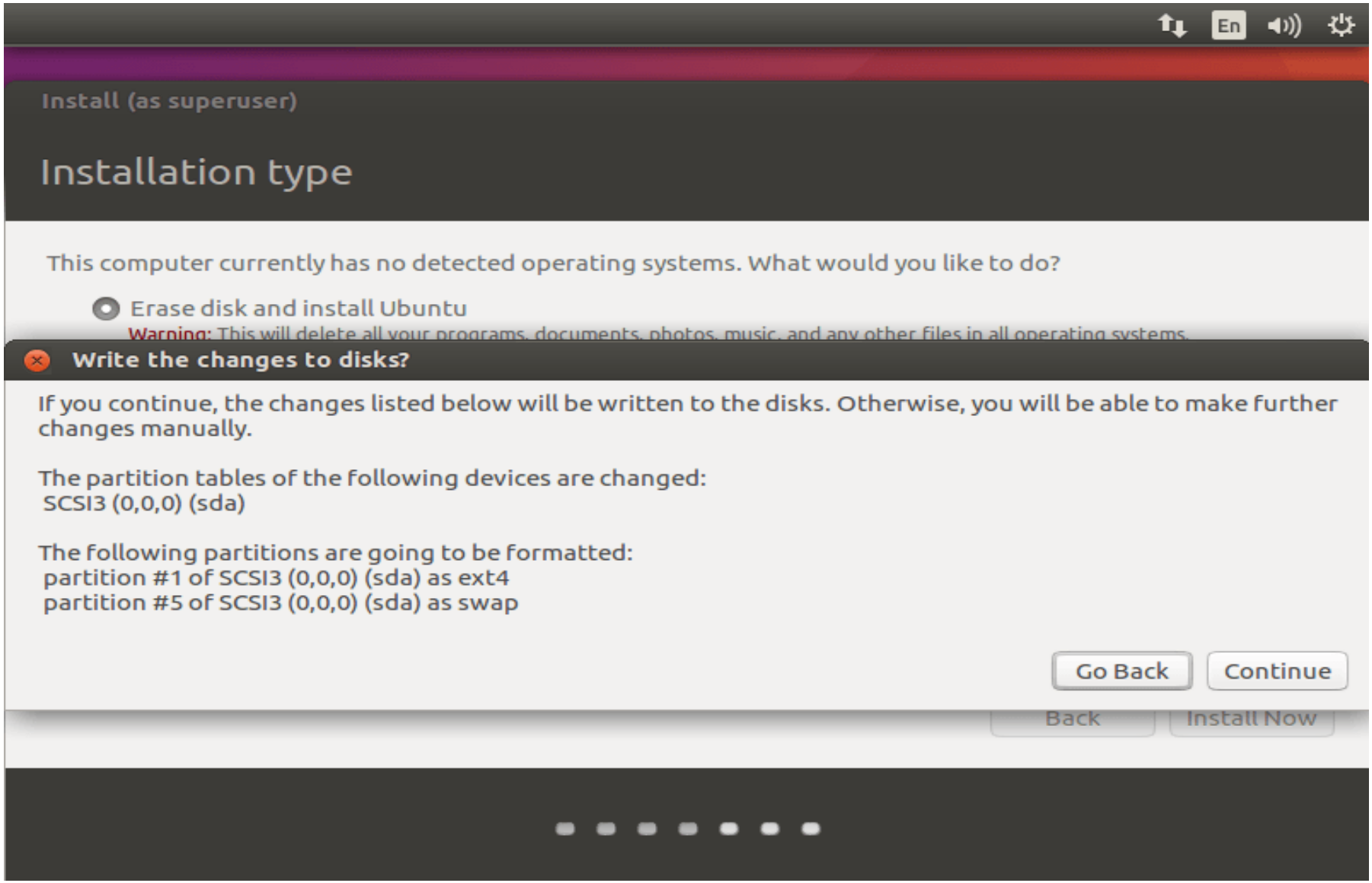


- Next is the installation type, this installation on the fresh HDD so I have only two option in the installation type. Depend on the other OS on your HDD you will get more options. **Please chose any one of the methods.**

  1. Erase disk and install Ubuntu (i.e. **it will format the entire drive and install the OS**). If you don't have any idea about the partitioning scheme simply click on Install Now.

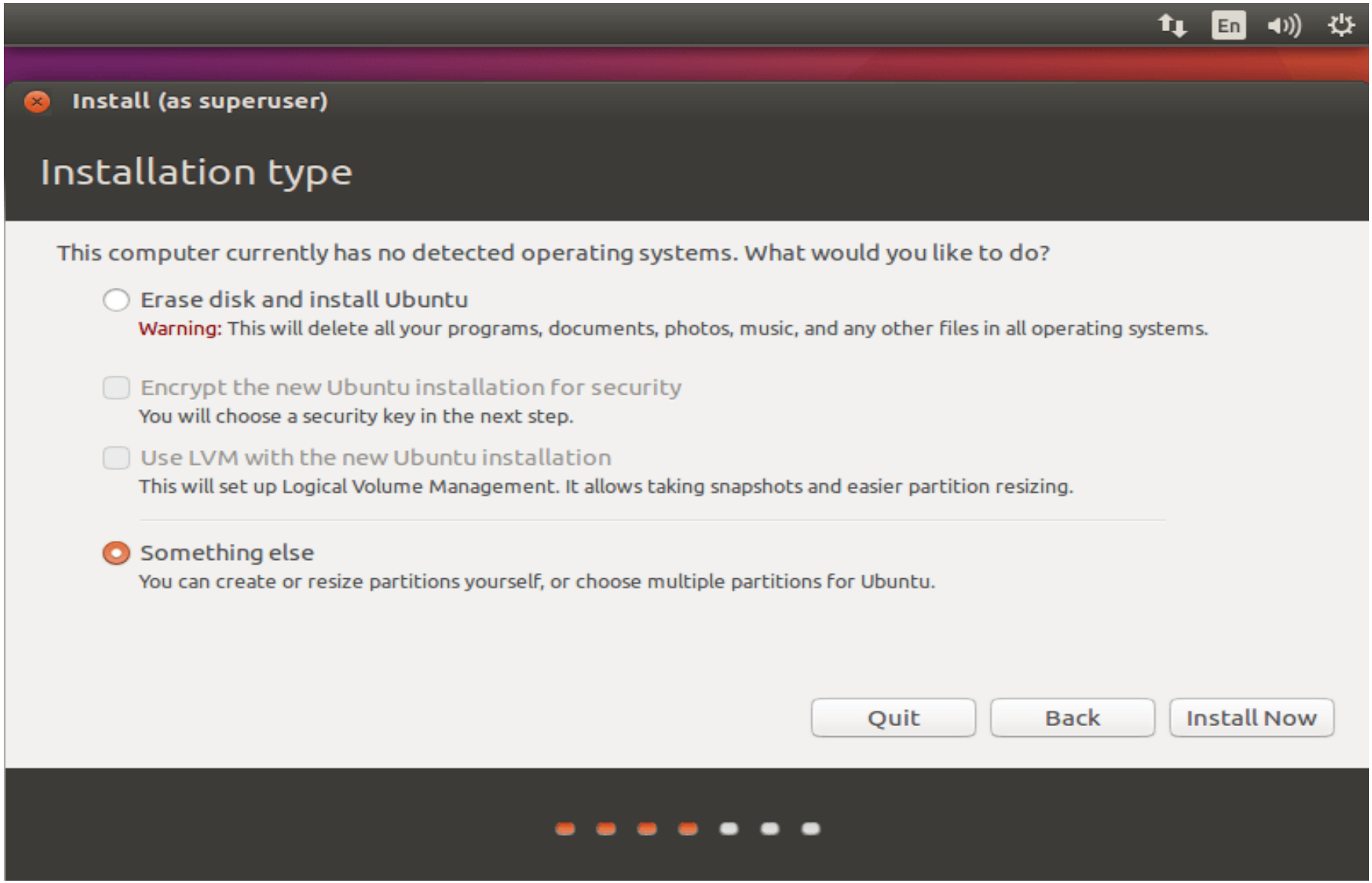
Once you clicked on Install Now, the installer will ask you to confirm the auto partitioning. Click on continue.




En
Install (as superuser)
Installation type
This computer currently has no detected operating systems. What would you like to do?
Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.
Write the changes to disks?
If you continue, the changes listed below will be written to the disks. Otherwise, you will be able to make further changes manually.
The partition tables of the following devices are changed:
SCSI3 (0,0,0) (sda)
The following partitions are going to be formatted:
partition #1 of SCSI3 (0,0,0) (sda) as ext4
partition #5 of SCSI3 (0,0,0) (sda) as swap
Go Back
Continue
Back
Install Now

2. Something else (i.e. **you can manually create the partition and install Ubuntu on your selected partition**), use this advanced mode if you are comfortable in partitioning your drives manually. Click on continue.

- Select something else and click install now



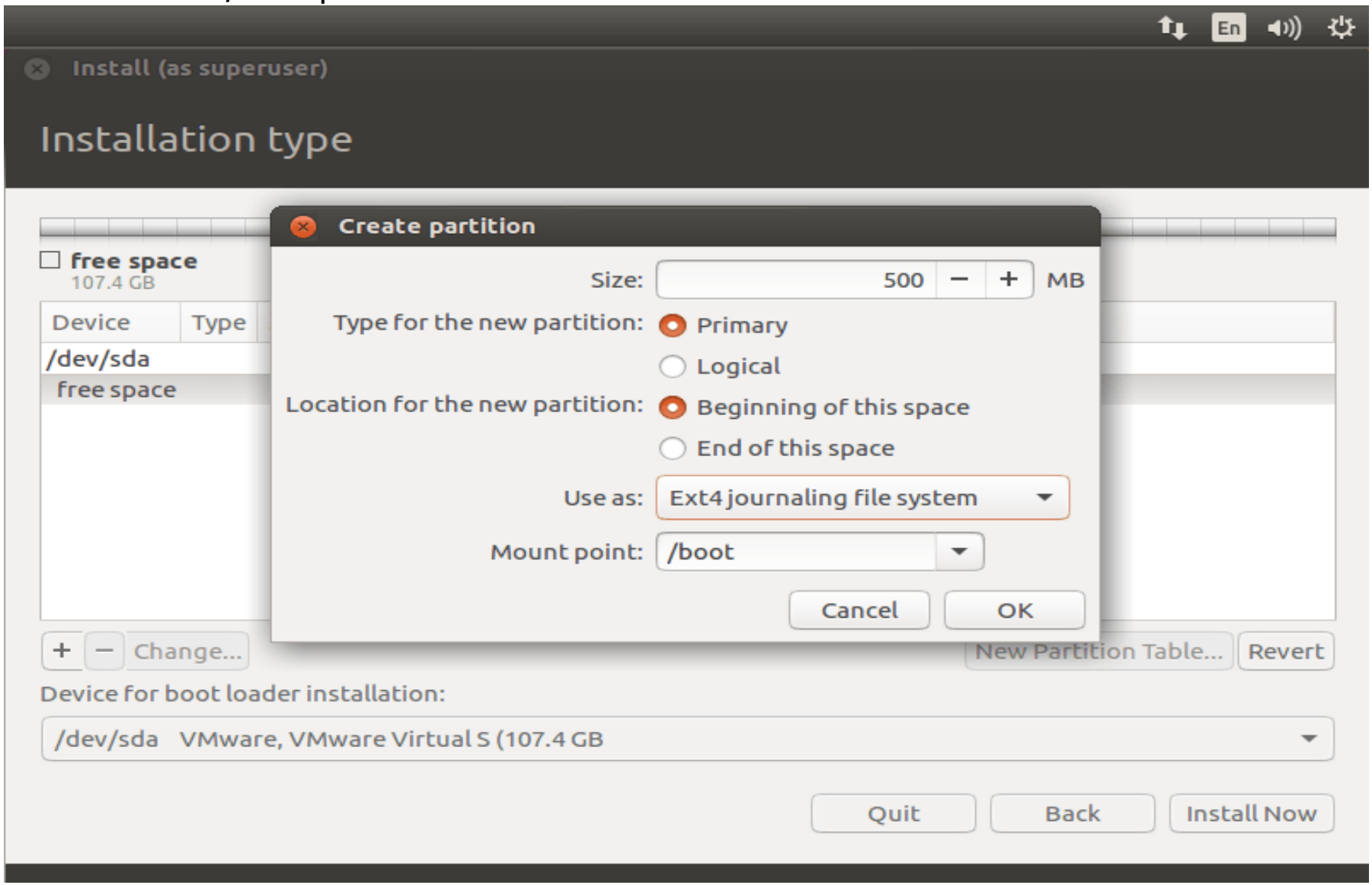
- Once you clicked, you would get the following page where installer lists available hard disk. In my case I have one hard disk size of 80GB, to create a partition; click on New Partition Table to create an empty partition.

- Since this is a new hard disk. Pop up will ask you to confirm, click on continue.

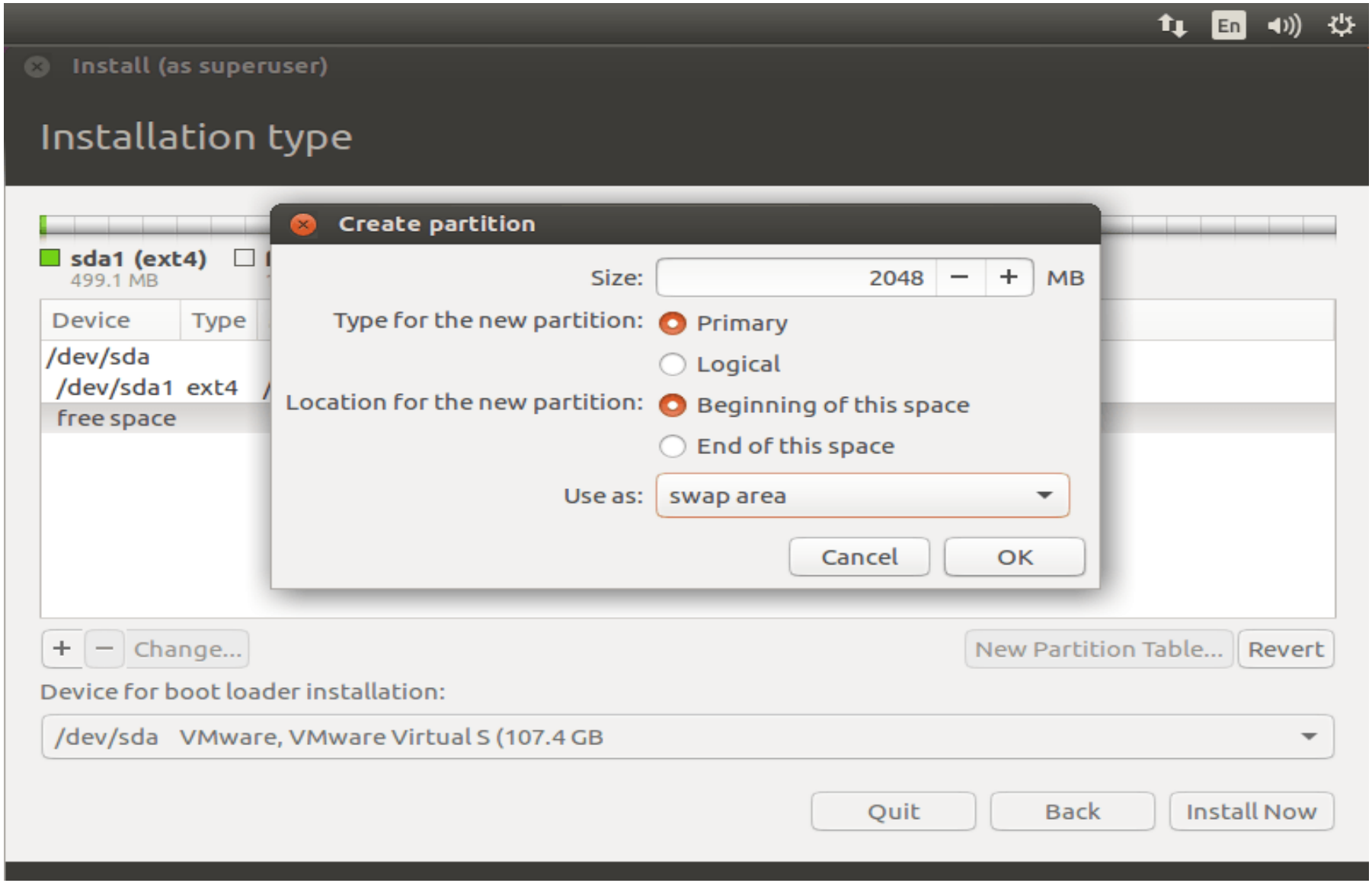
- Partition scheme will be like below:

  - **/boot – 500MB**
  - **swap – 2048MB**
  - **/ – Remaining (99GB)**

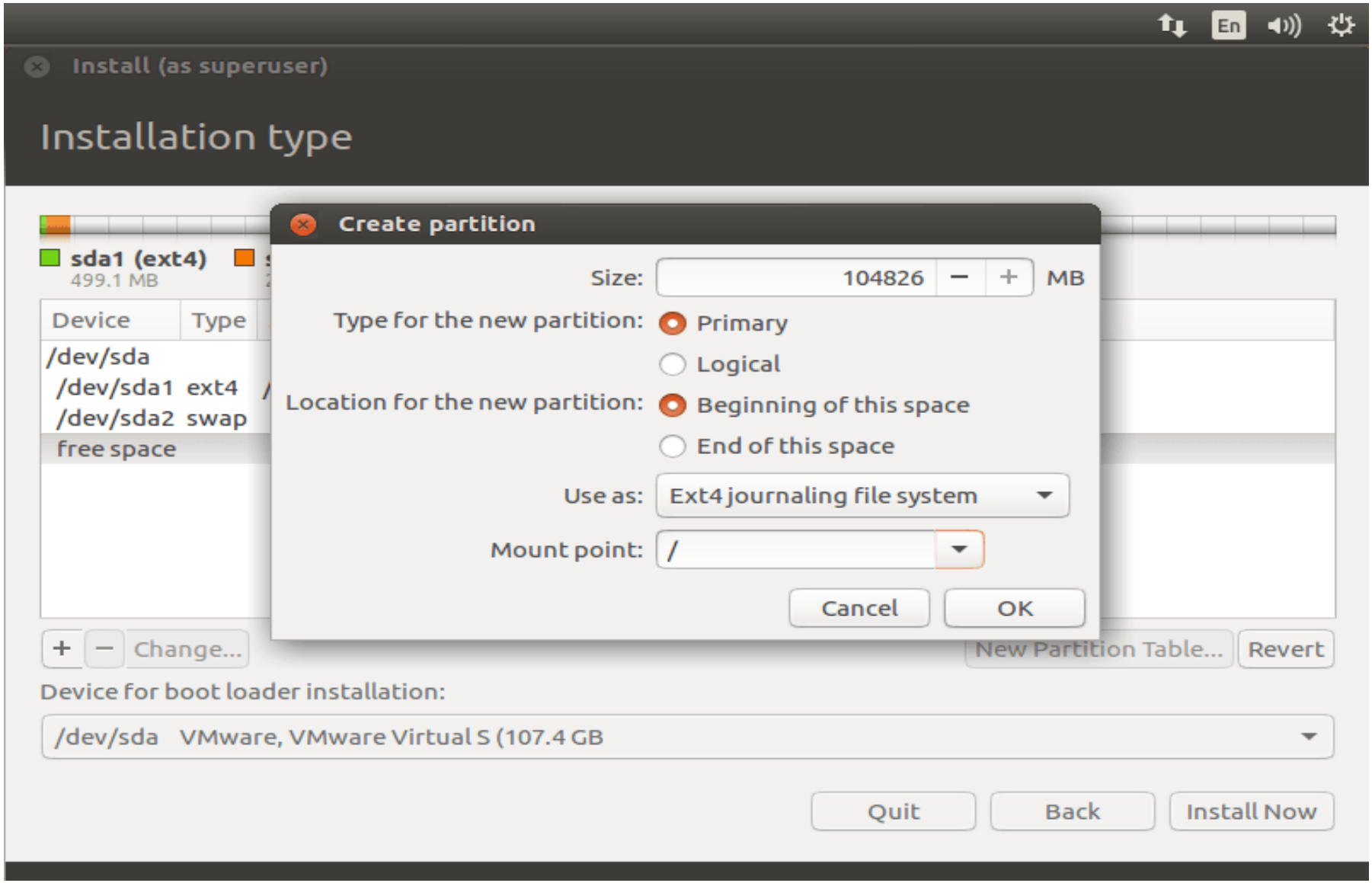
- Select free space and click on the **+ sign** at the bottom to create partitions. Following shows for /boot partition.



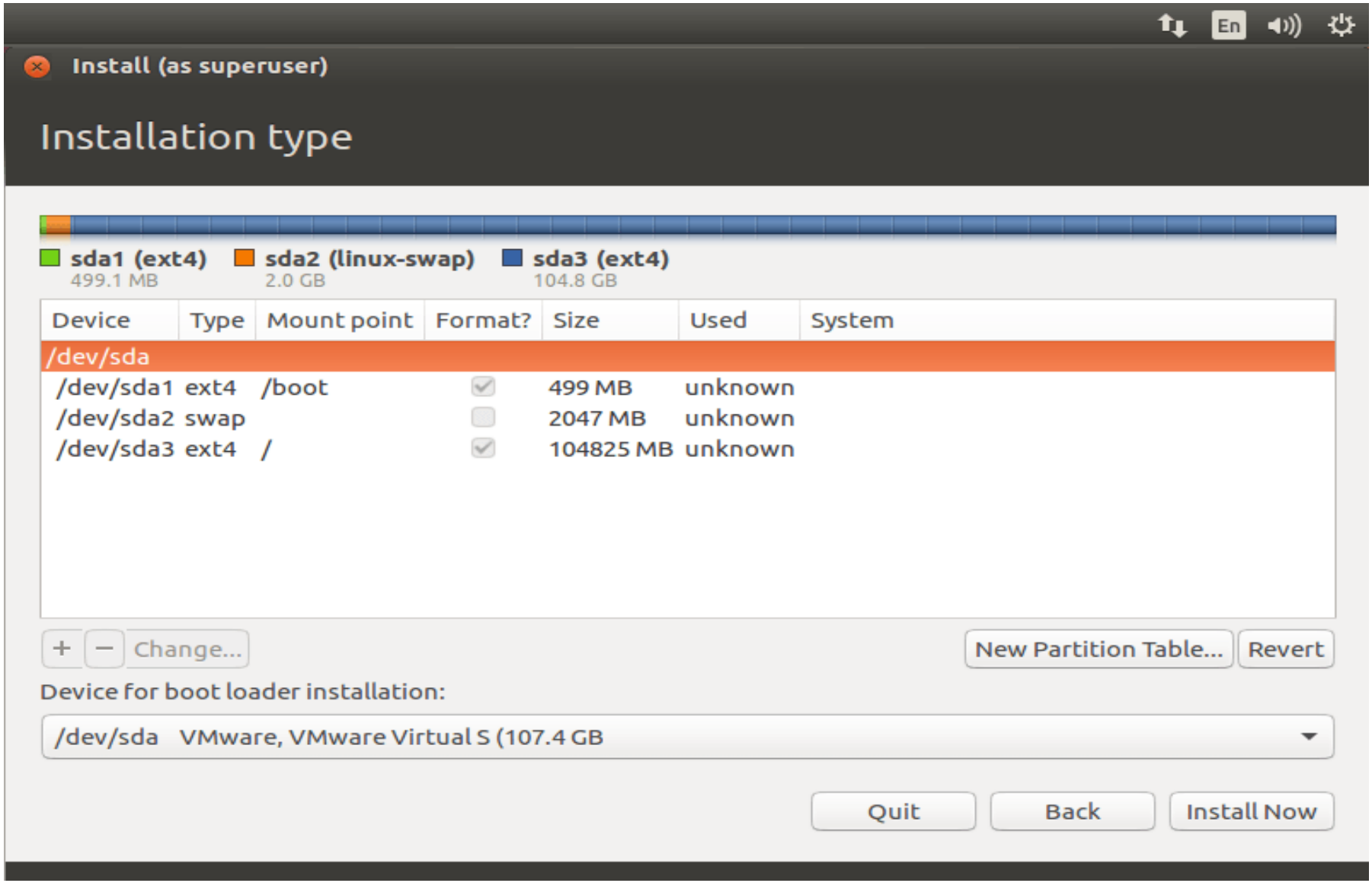
Following screen show for the swap partition, it is important to select **use as swap area**



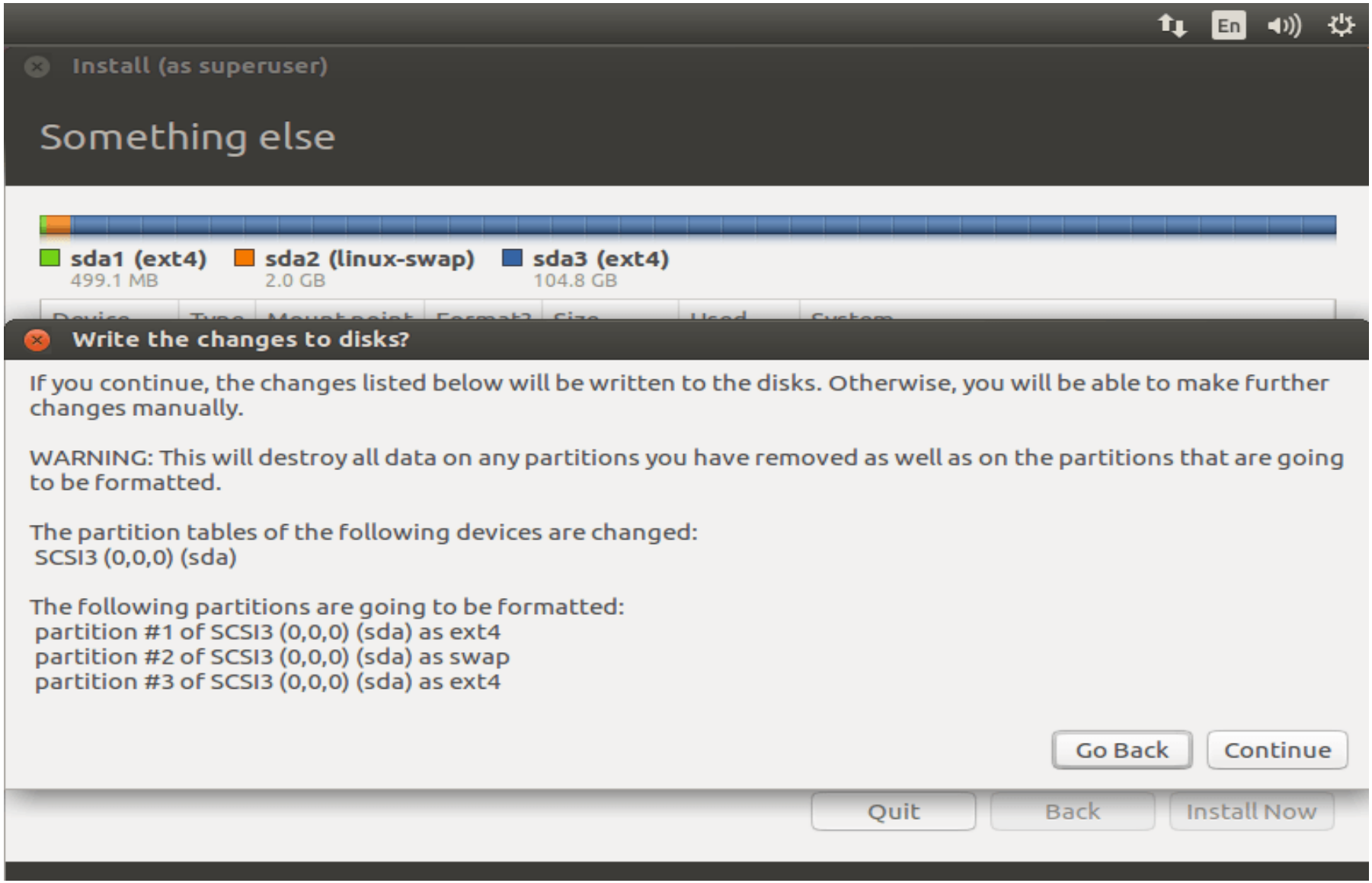
Following is for / (root) partition.



Review your partition layout and click on install now.

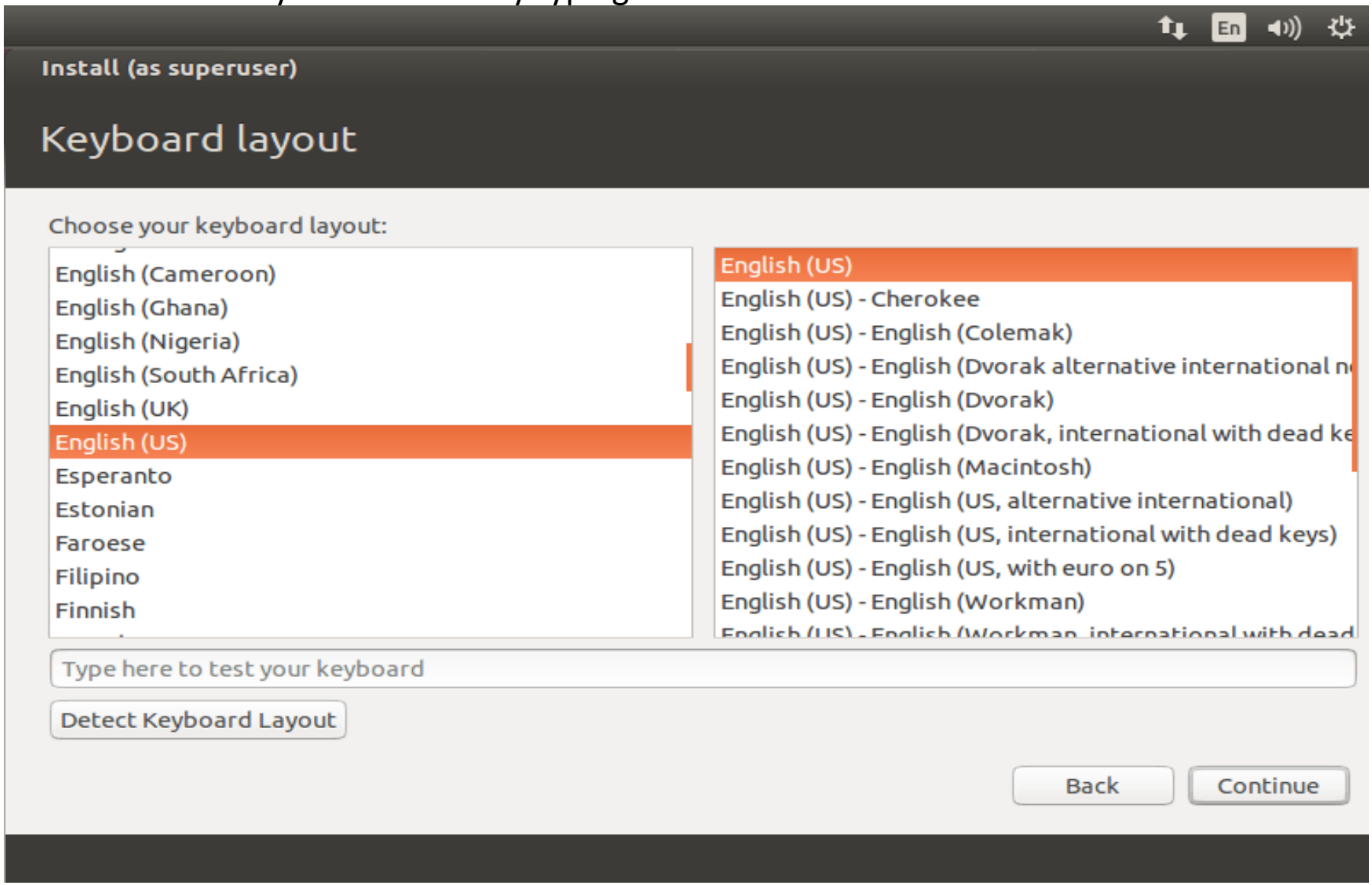


Write the changes to disk by clicking on continue.



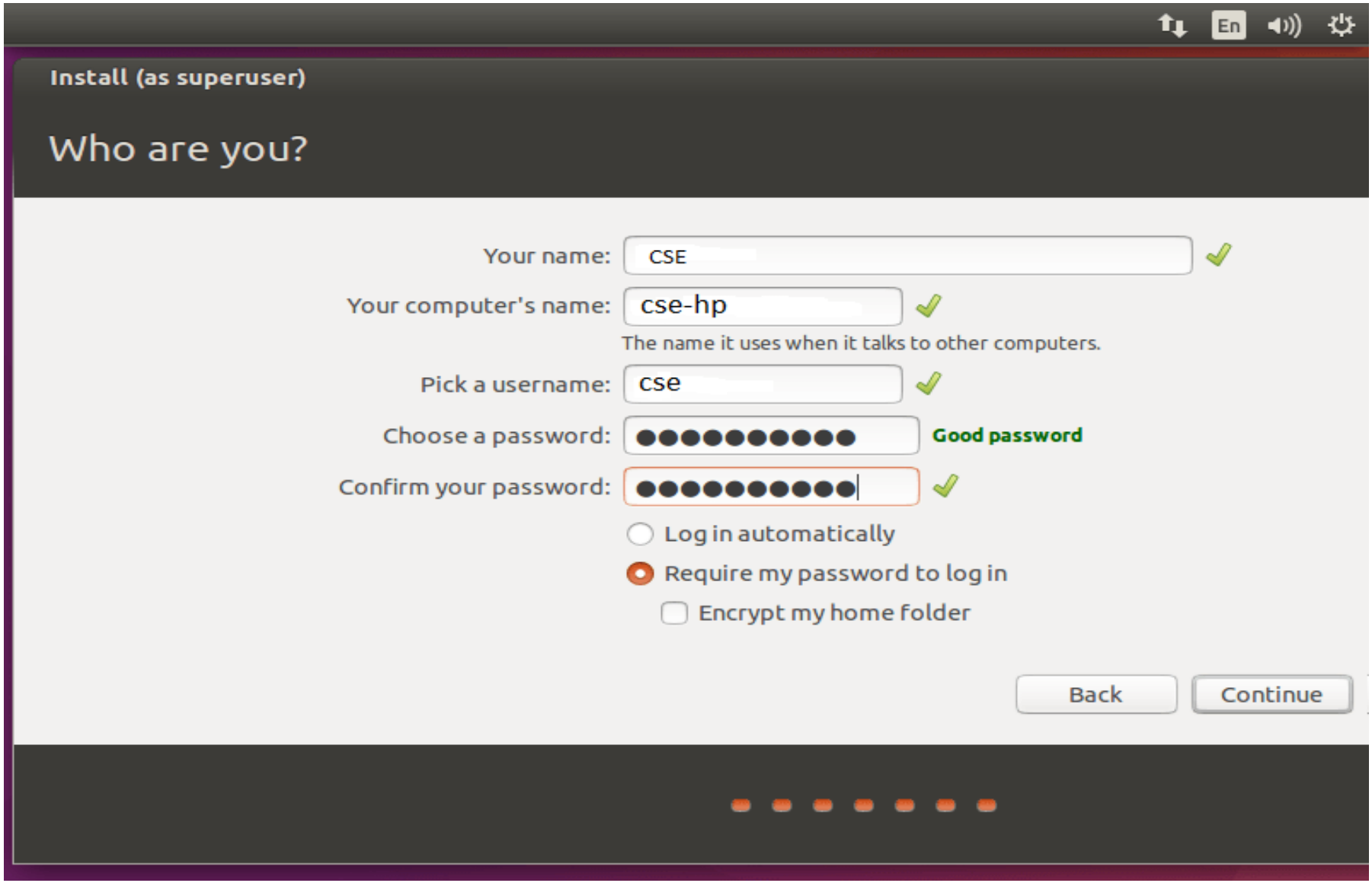
Select your location next screen.

Select your keyboard layout. If you are not sure, use the '**Detect Keyboard Layout**' option. You can also test your selection by typing in the test text box.

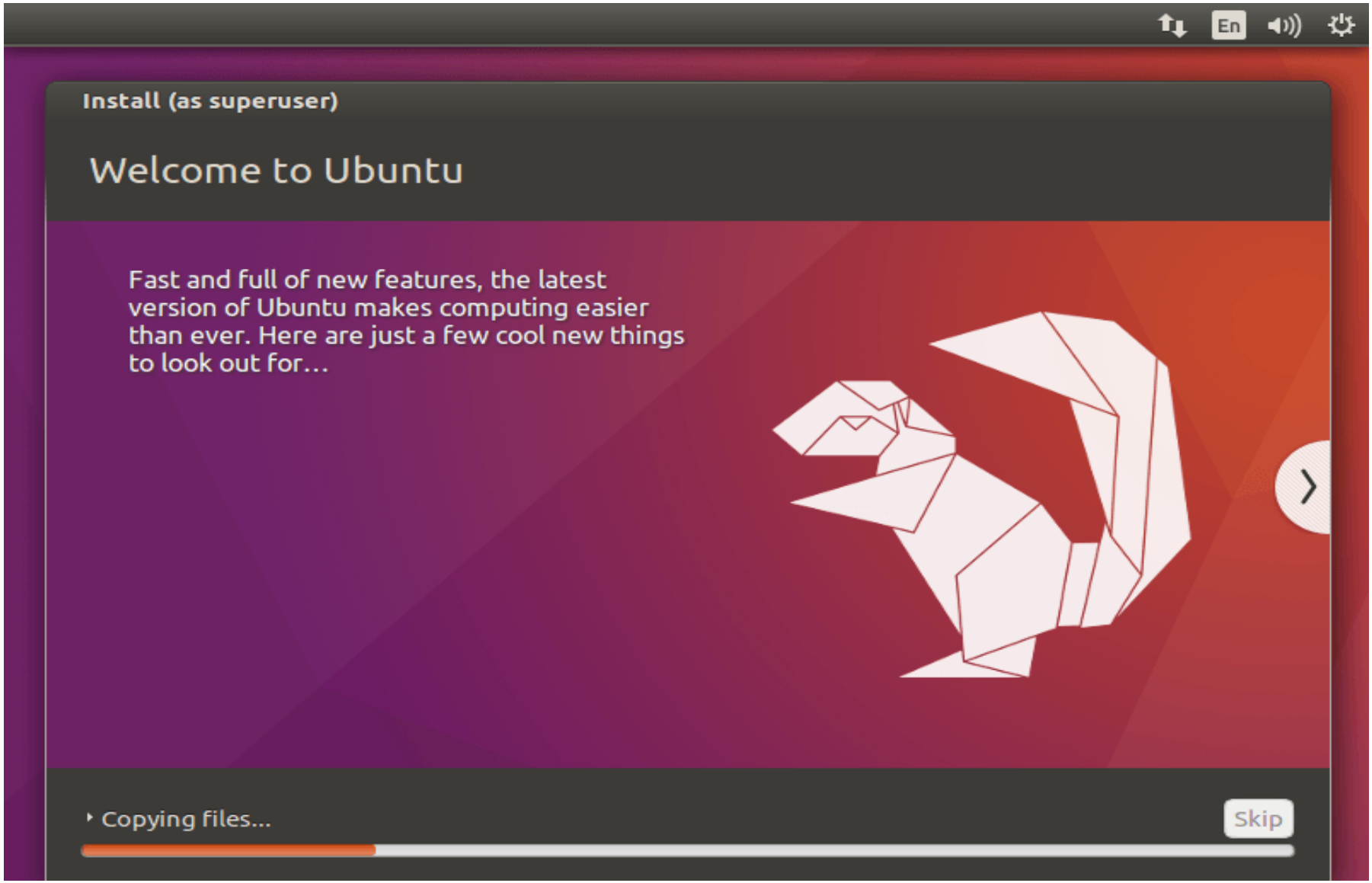


- In the final screen of the installation wizard, you will be prompted to enter information about the user that you wanted to create on the system. Enter your information in this screen.
- Here is one thing you should remember – if you select '**Login automatically**', System will directly take you to desktop without asking your credentials.
- It's best if you give a **very secure password** for your installation. Ubuntu will tell you whether your password is secure or not.
- If you select '**Encrypt my home folder**' it will make all the files and folders in your home folder more secure from unauthorized viewing if you have multiple users using your computer. When you log into your computer your files are seamlessly decrypted for just your session.
- If you are not sure, leave this box unchecked.

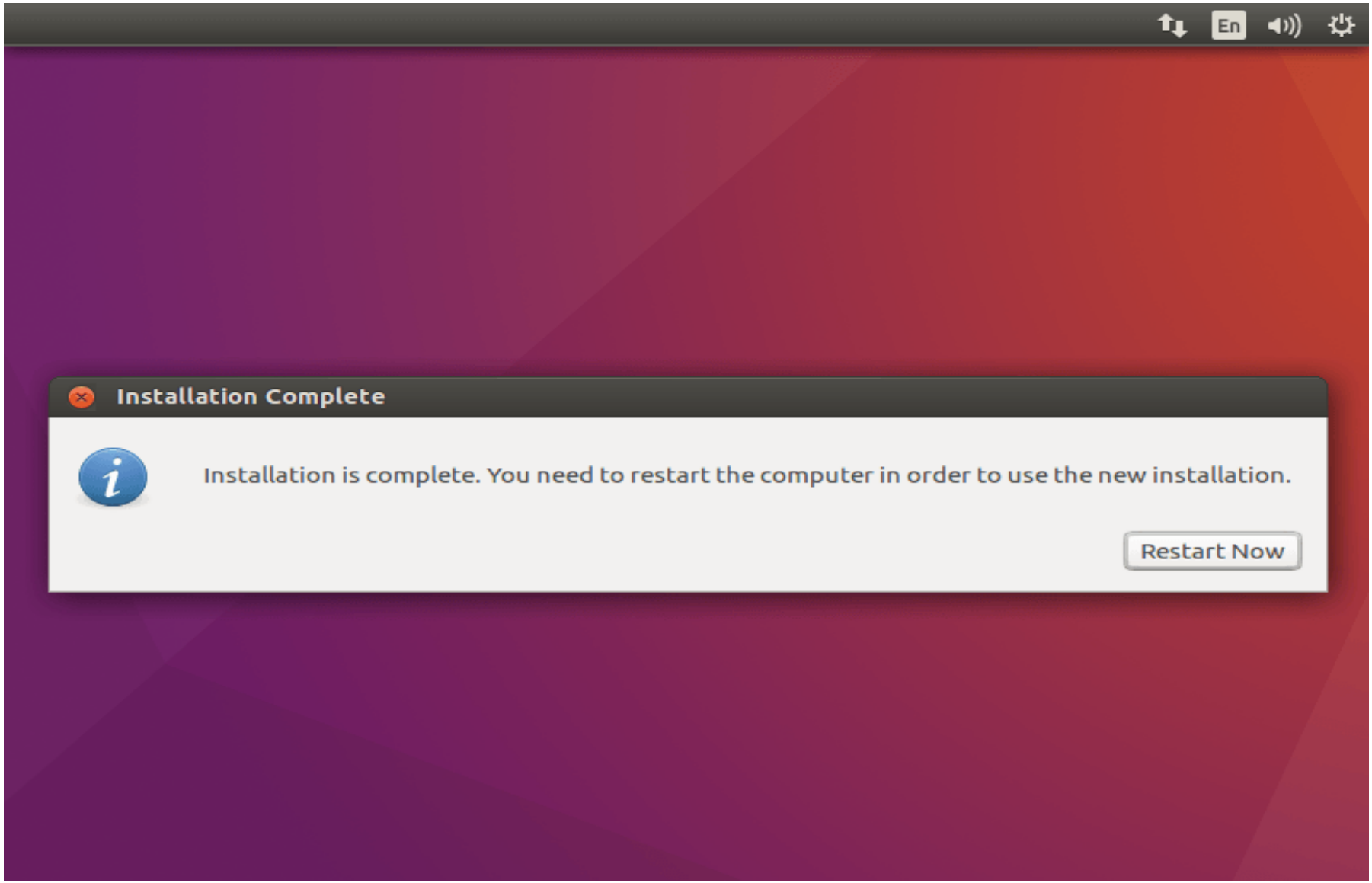
Once it’s done, click on continue.



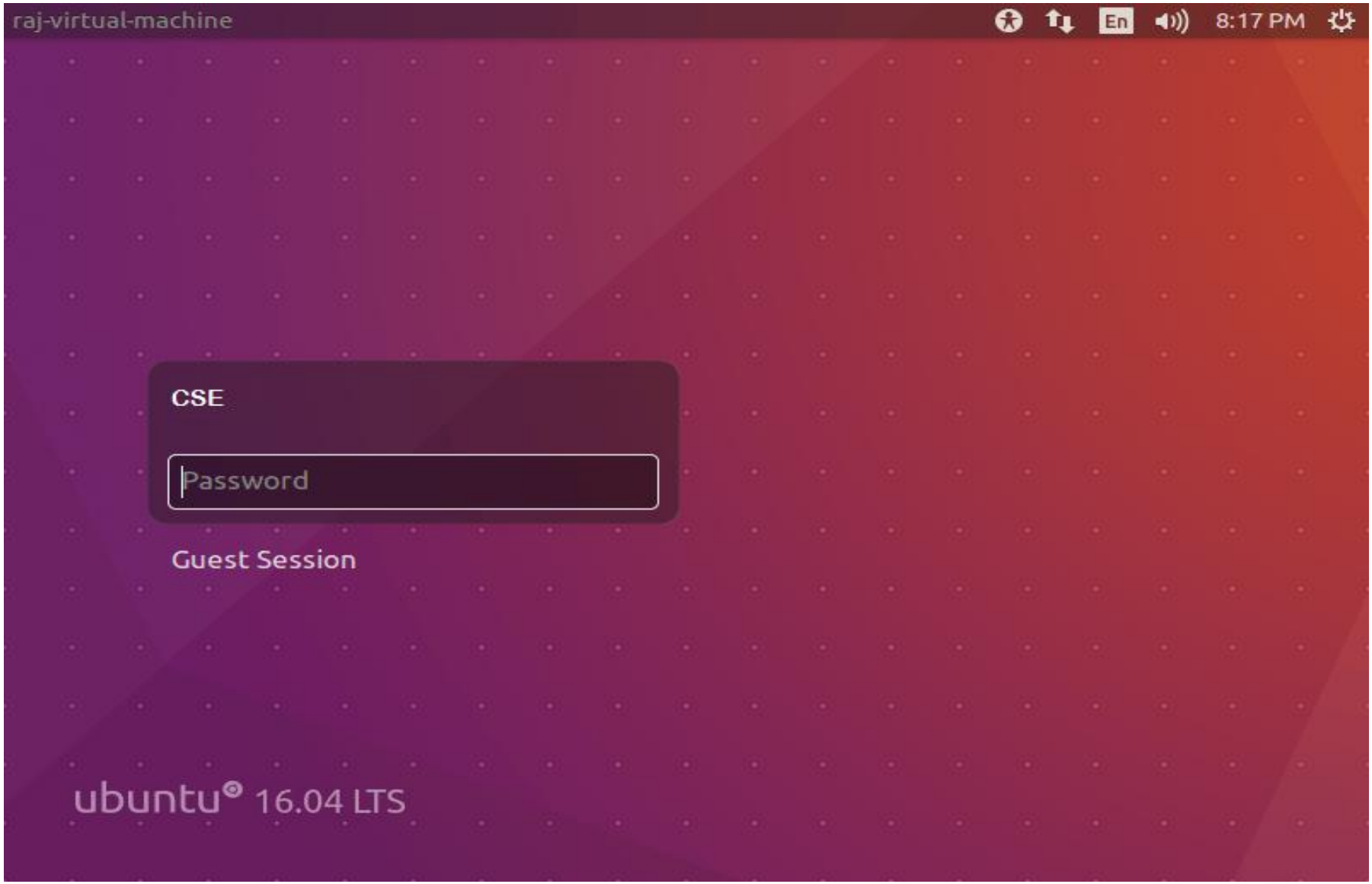
Below screenshot shows installing Ubuntu 16.04



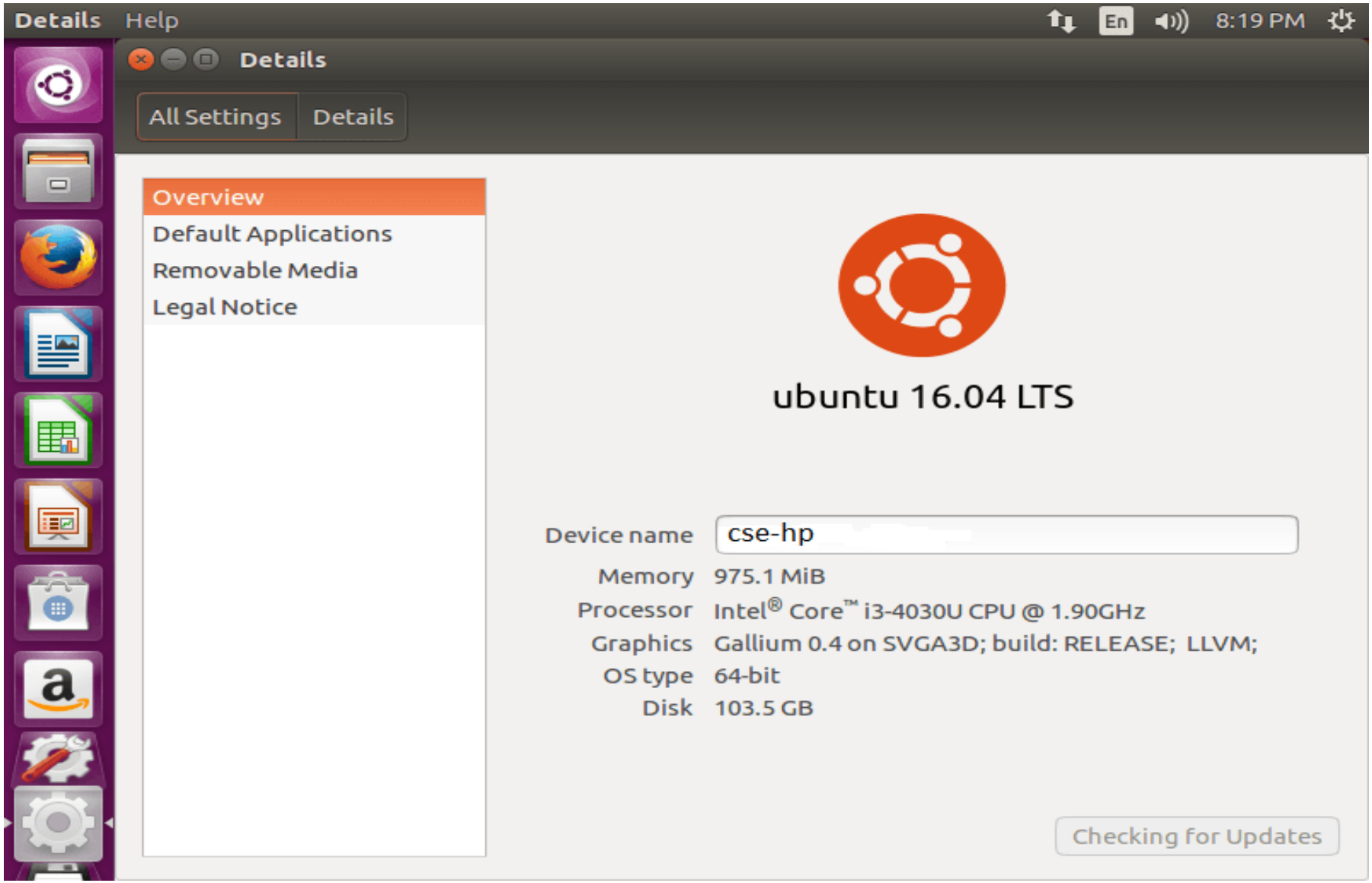
Once the installation is over, click on restart now



Once your machine is restarted, you will get a login window. Login with username and password that you created earlier



Check The System Details



Virtual Machine-OS

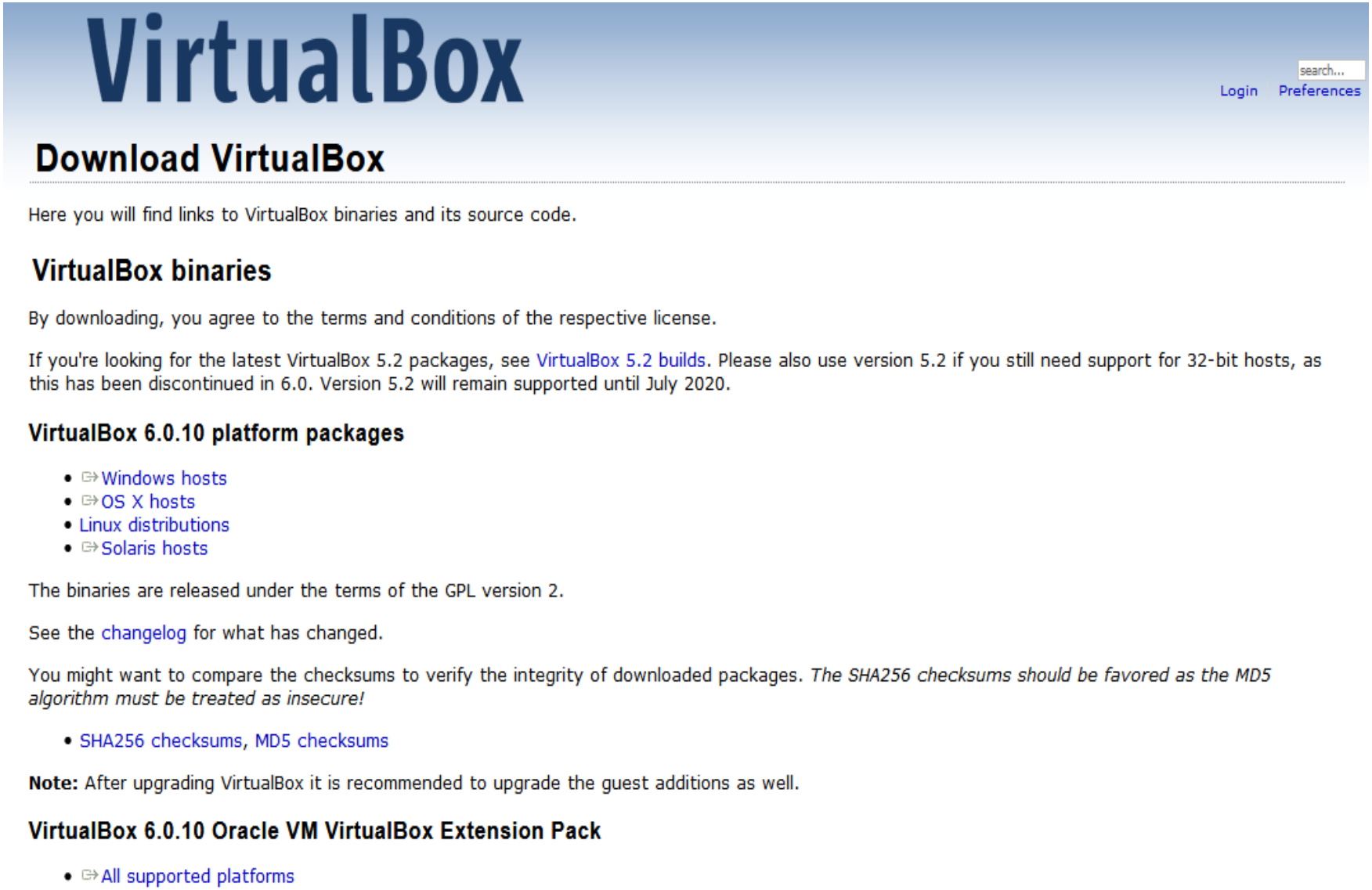
Virtual Box Installation

Downloading VirtualBox

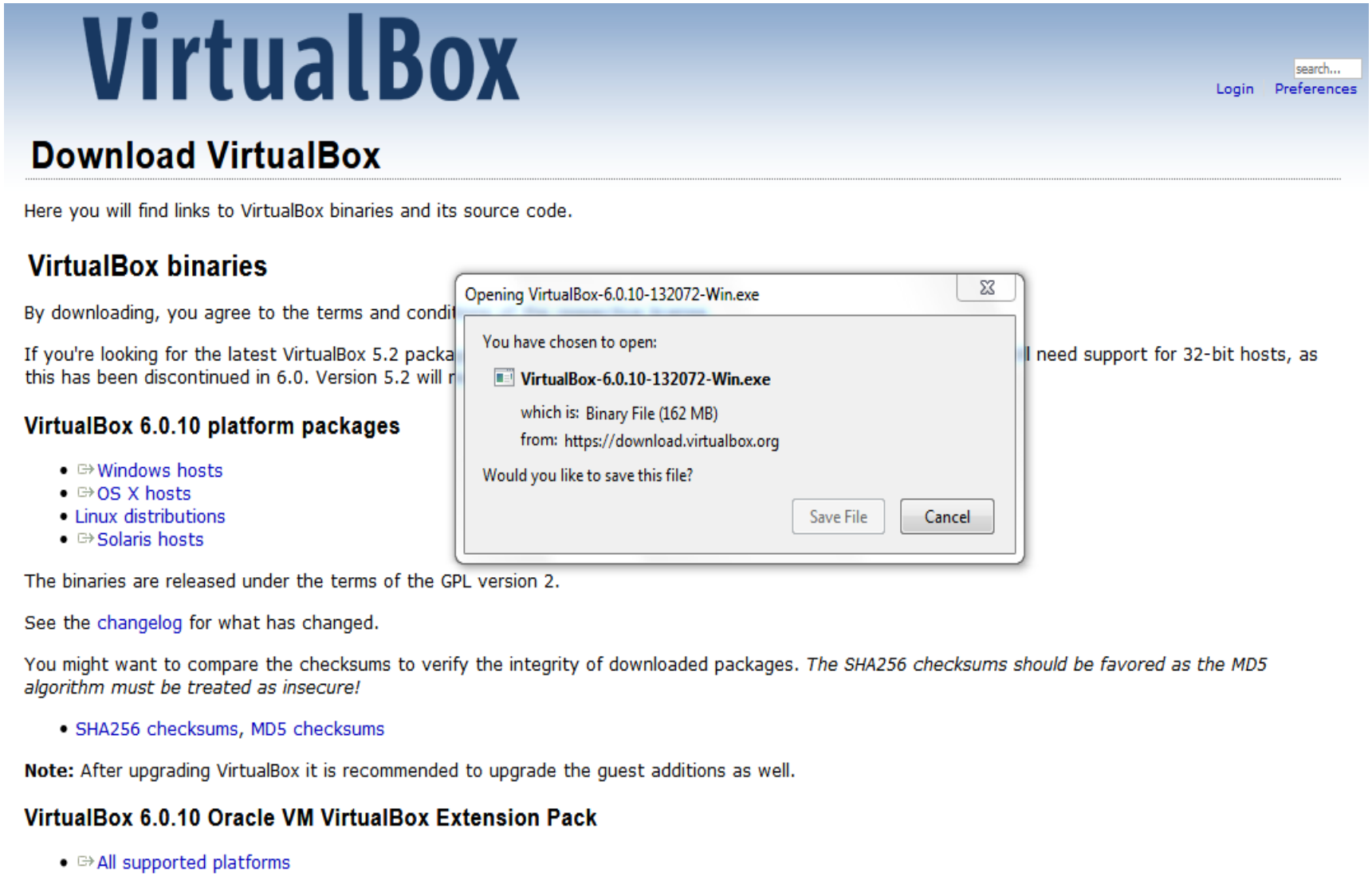
- Click the link to download VirtualBox
  - https://www.virtualbox.org/

- Click the Download VirtualBox 6.0 Button

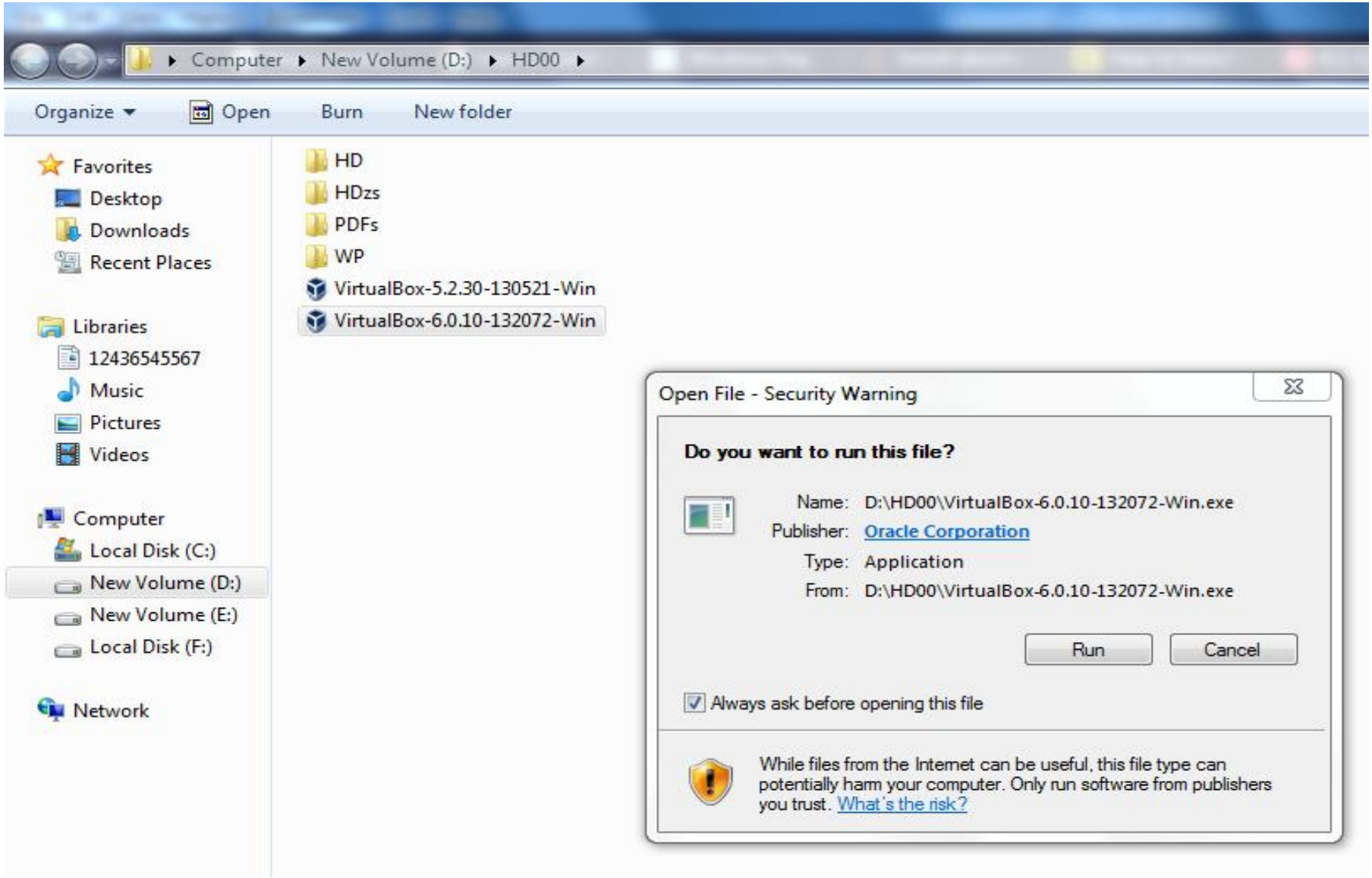
- Download VirtualBox to click Windows Hosts Link



- Click the Save File button to save the VirtualBox executable file

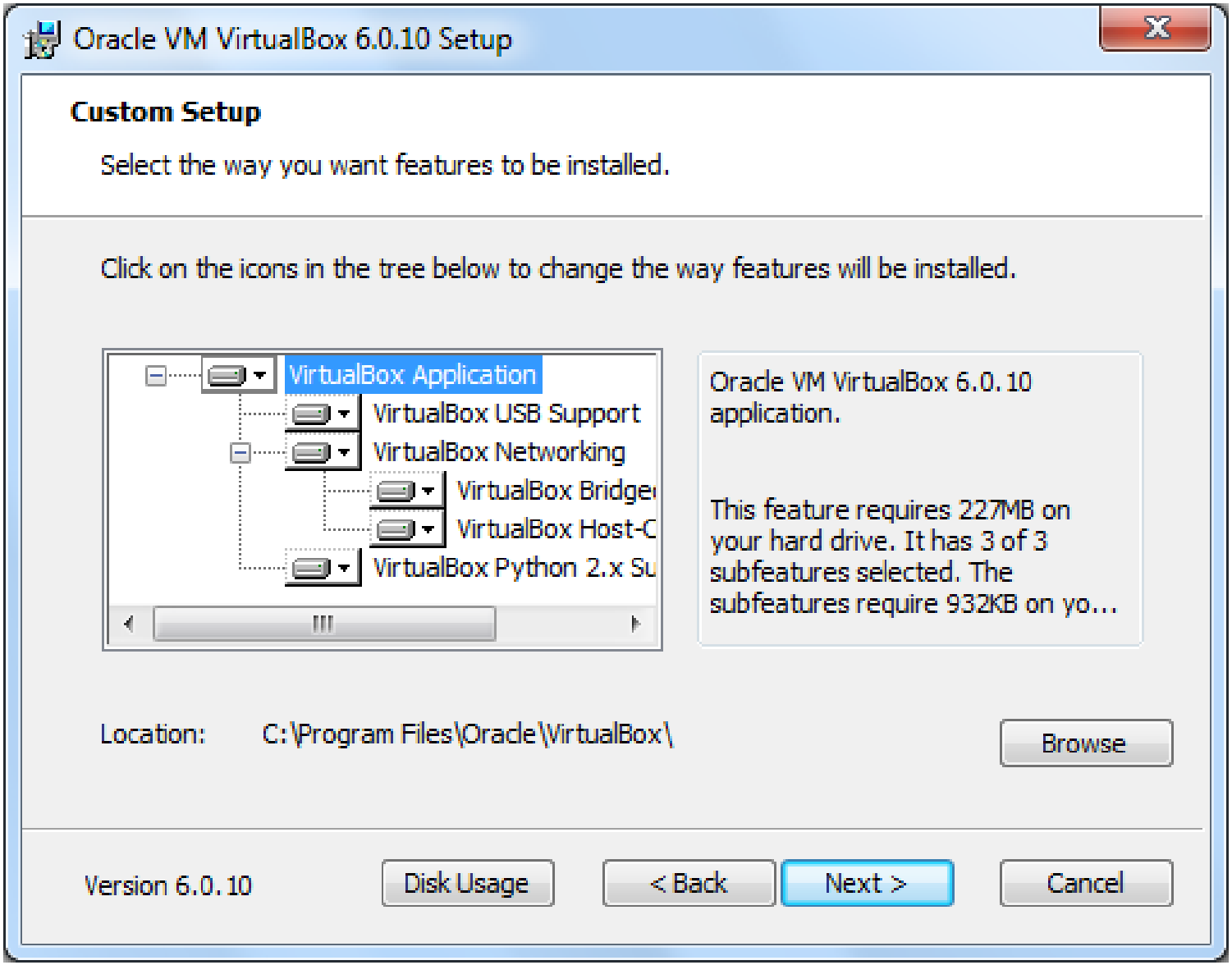


- Run the VirtualBox setup from the downloaded file location

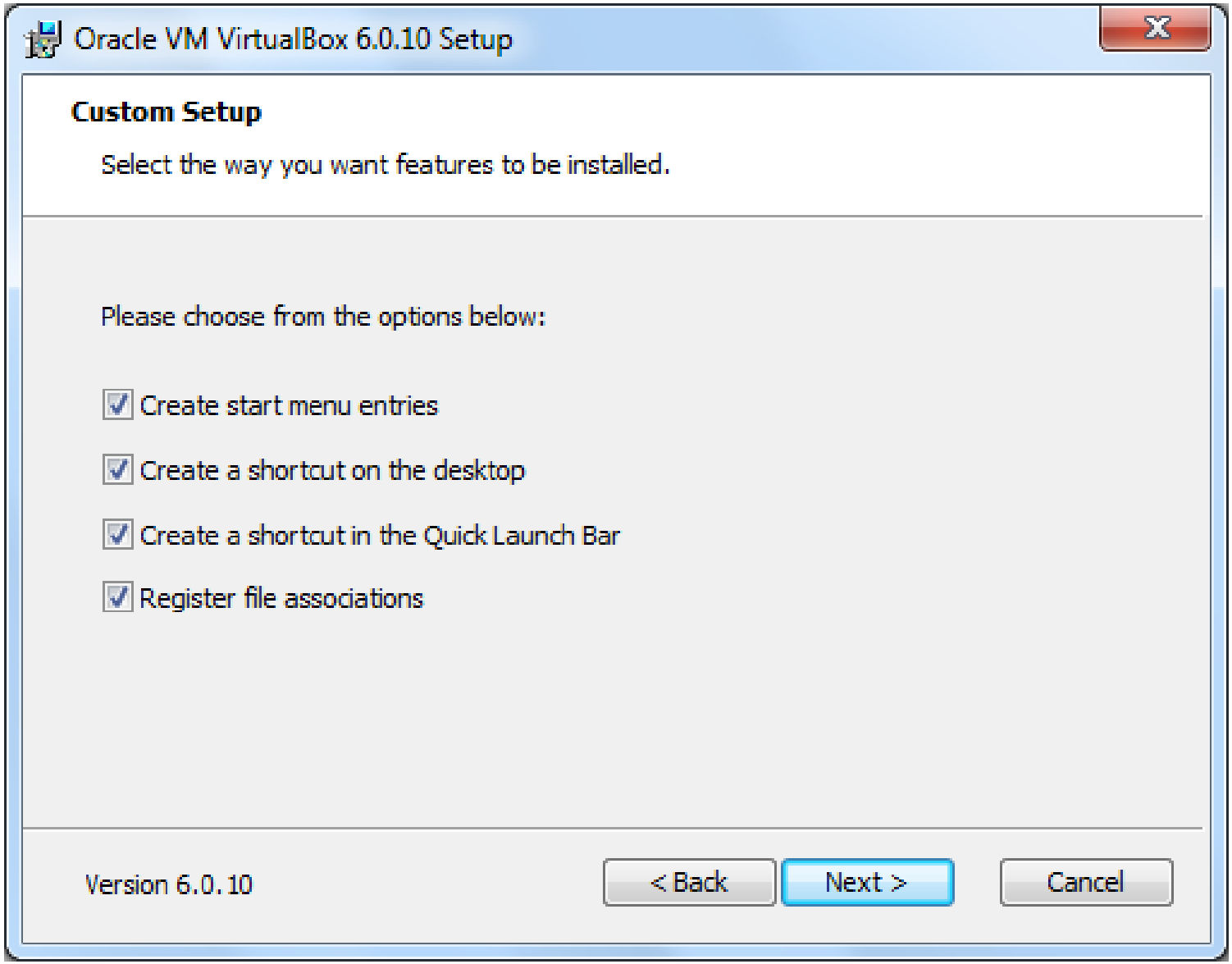


- Click on the next button

- Click on the next button

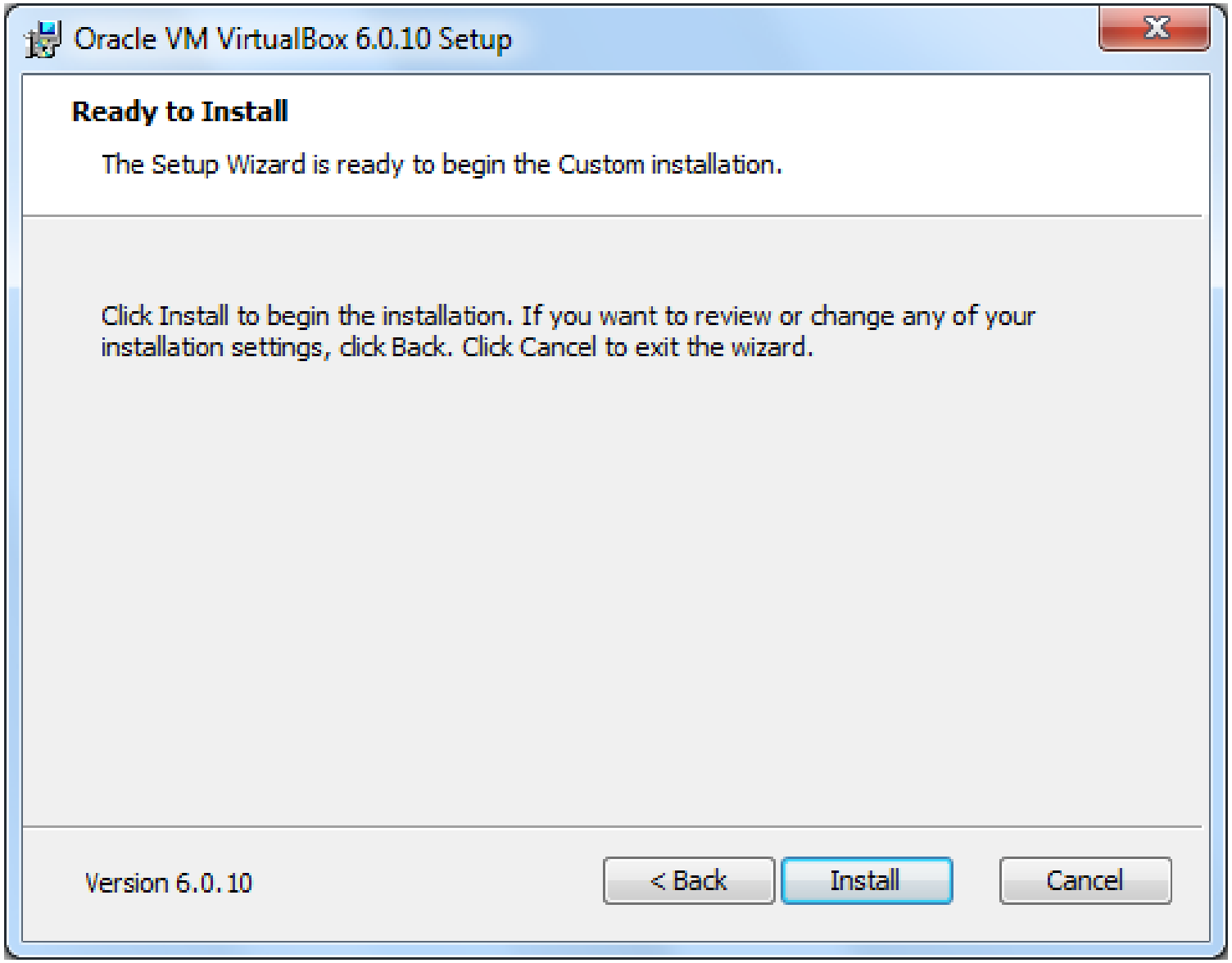


- Click on the next button

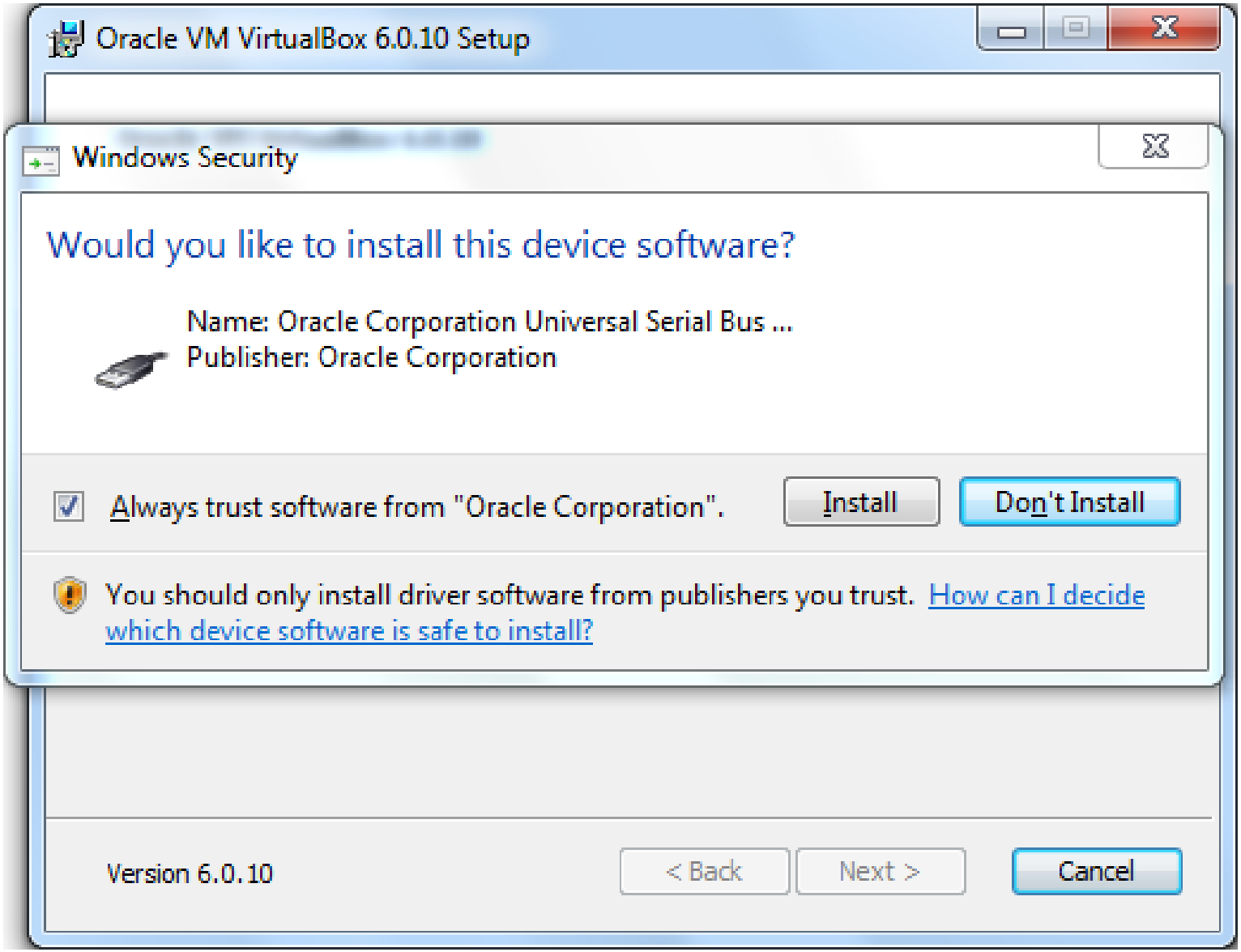


- Click on Yes Button

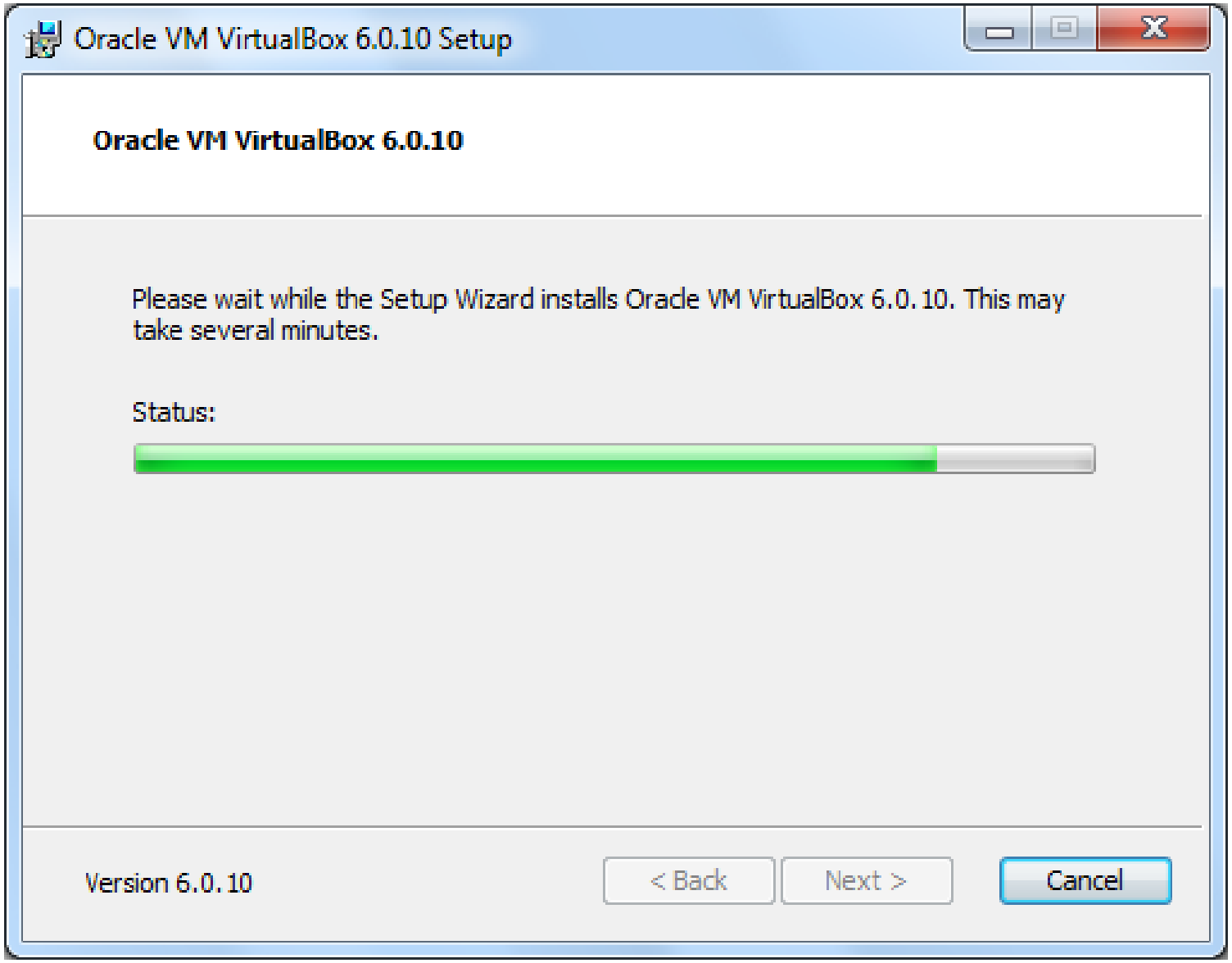
- Click on Install Button



- Select The Check Box and Click Install Button



- Installation Process going on

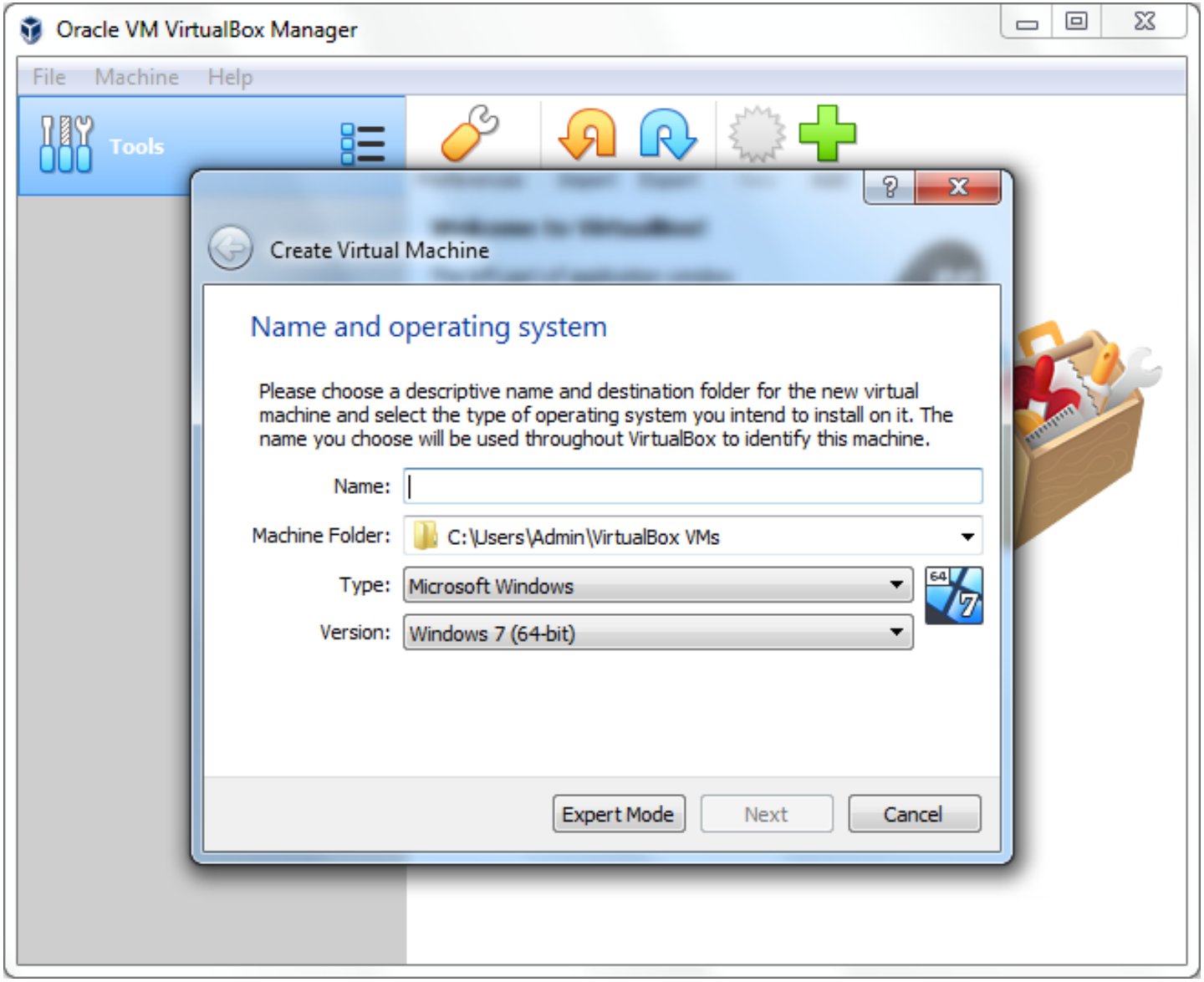


- Click the Finish Button Virtual Box Installation Completed

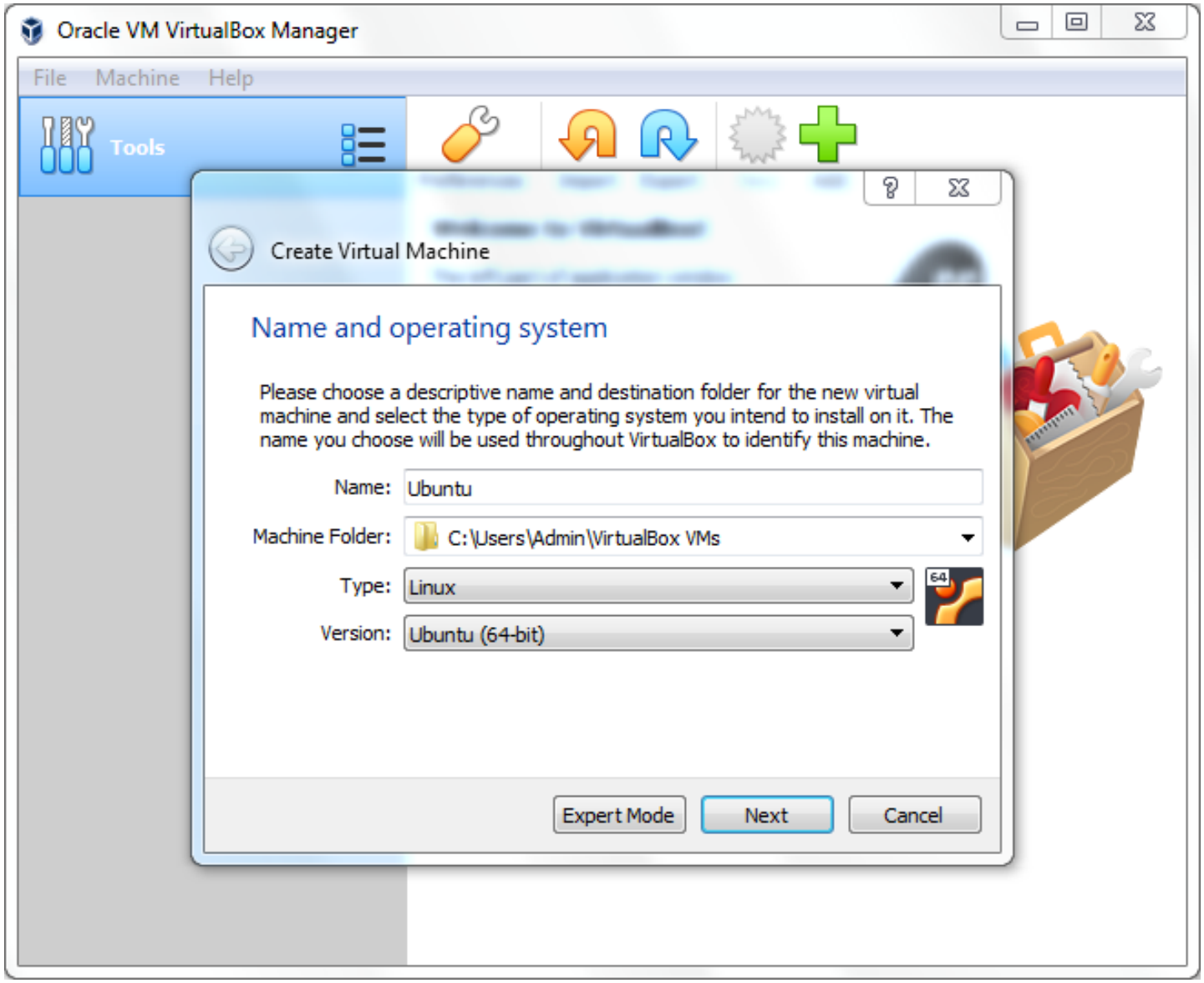
Virtual Box – Virtual Machine Creation

- Open Oracle Virtual Box Manager

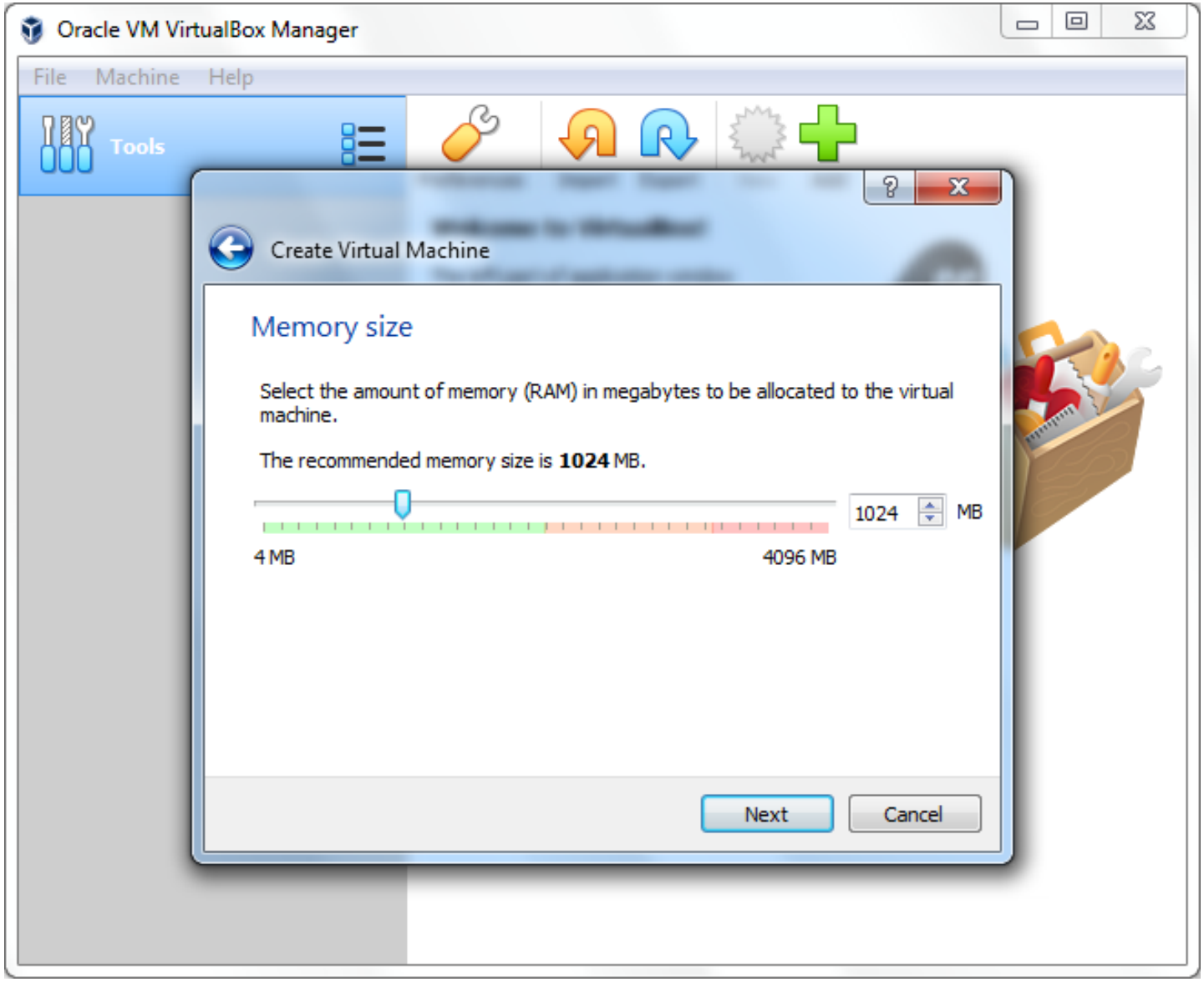
- Click **New**


Oracle VM VirtualBox Manager
File
Machine
Help
Tools
Create Virtual Machine
Name and operating system
Please choose a descriptive name and destination folder 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:
Machine Folder:
C:\Users\Admin\VirtualBox VMs
Type:
Microsoft Windows
Version:
Windows 7 (64-bit)
Expert Mode
Next
Cancel

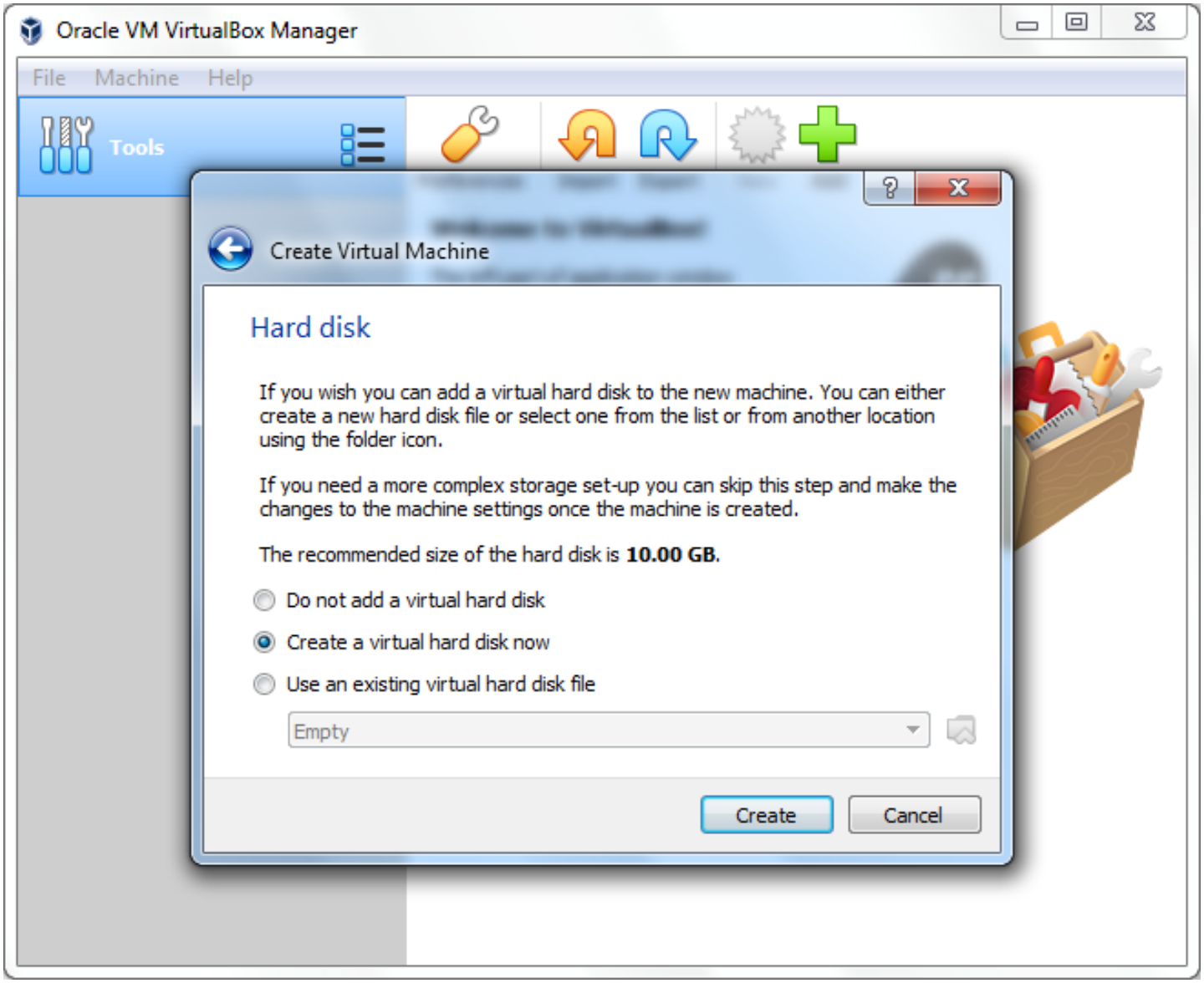
- Type the Virtual Machine Name and Version Then click Next button



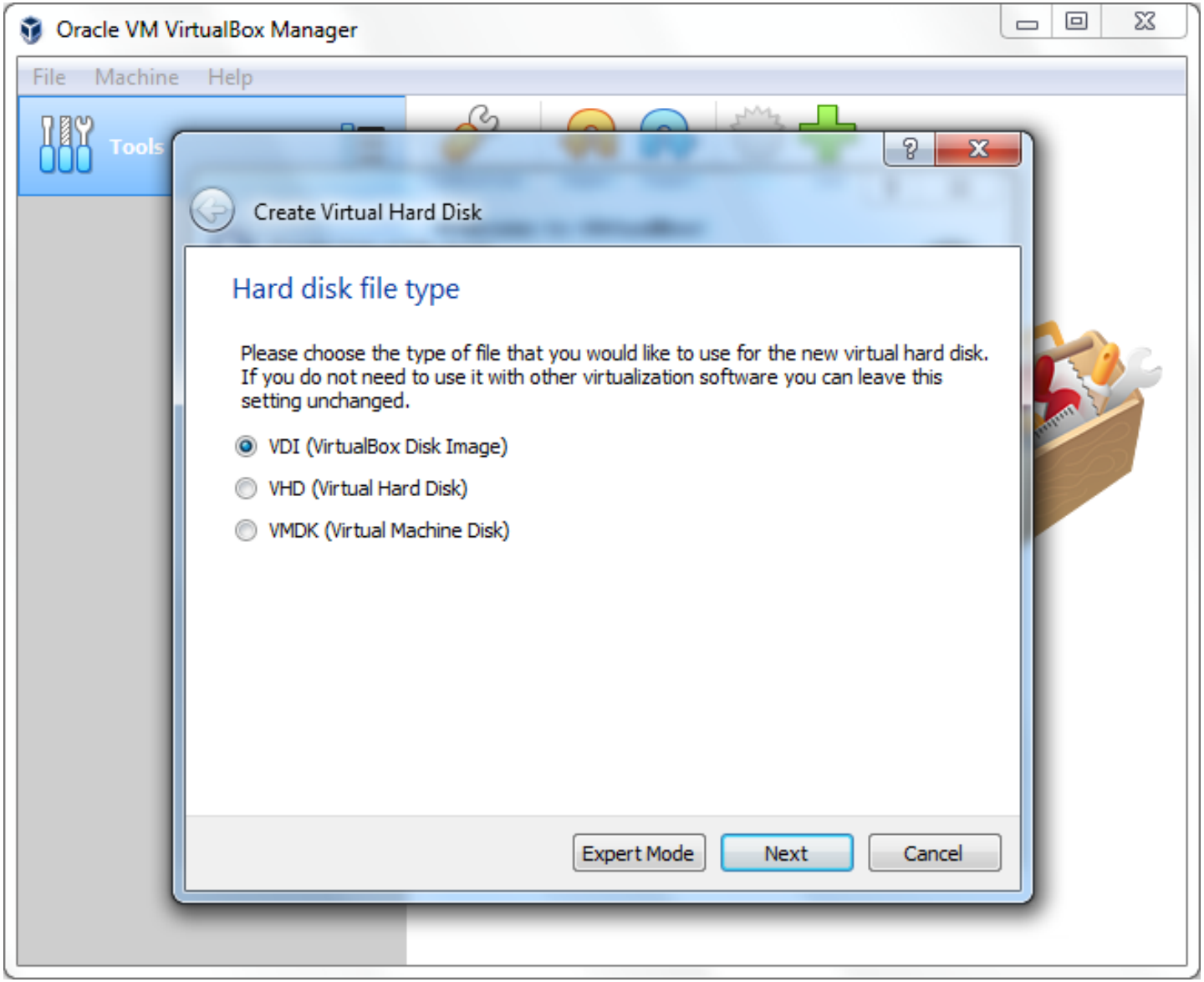
- Select the Memory size and click the Next button



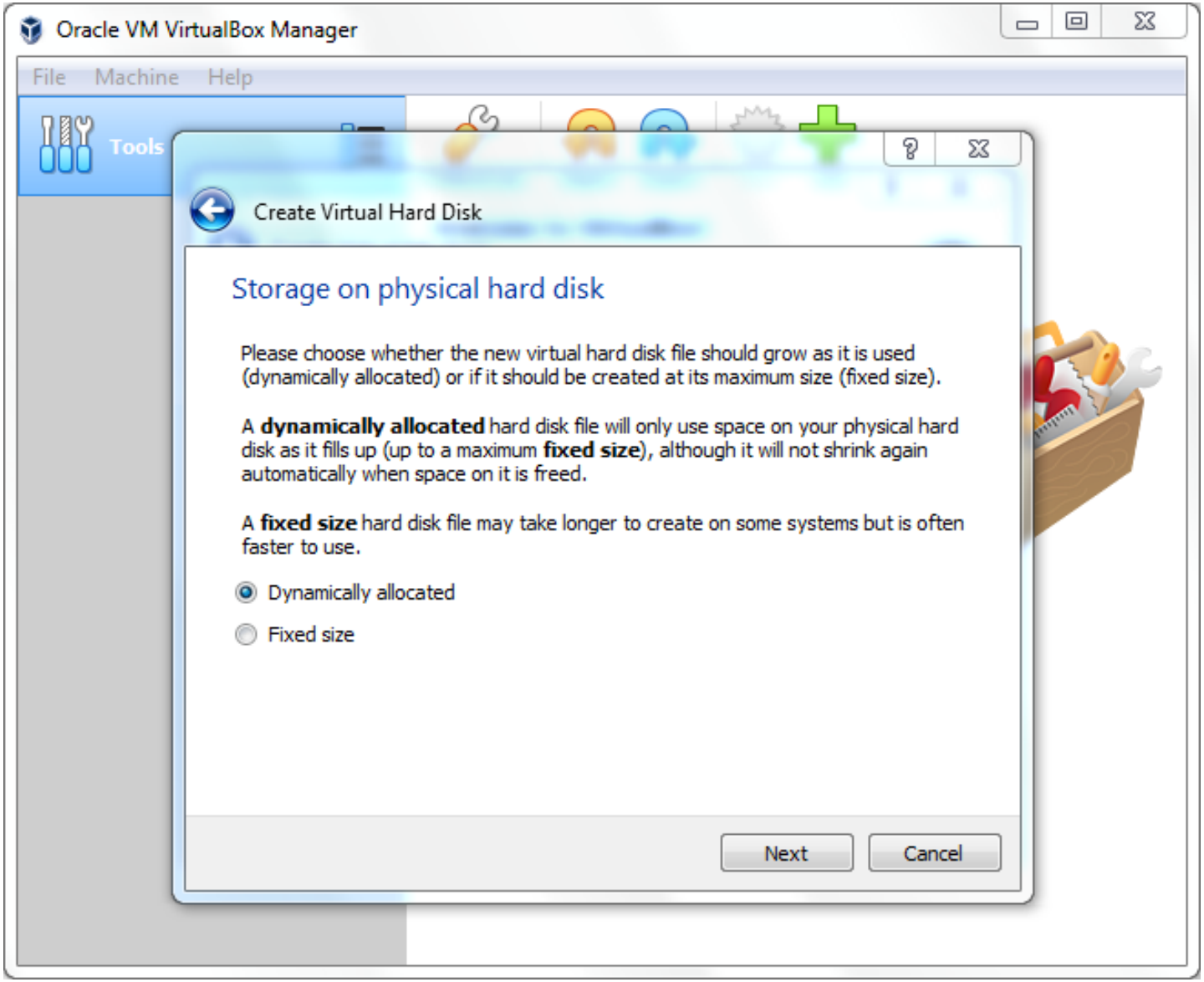
- Select Create the virtual hard disk now option then click create



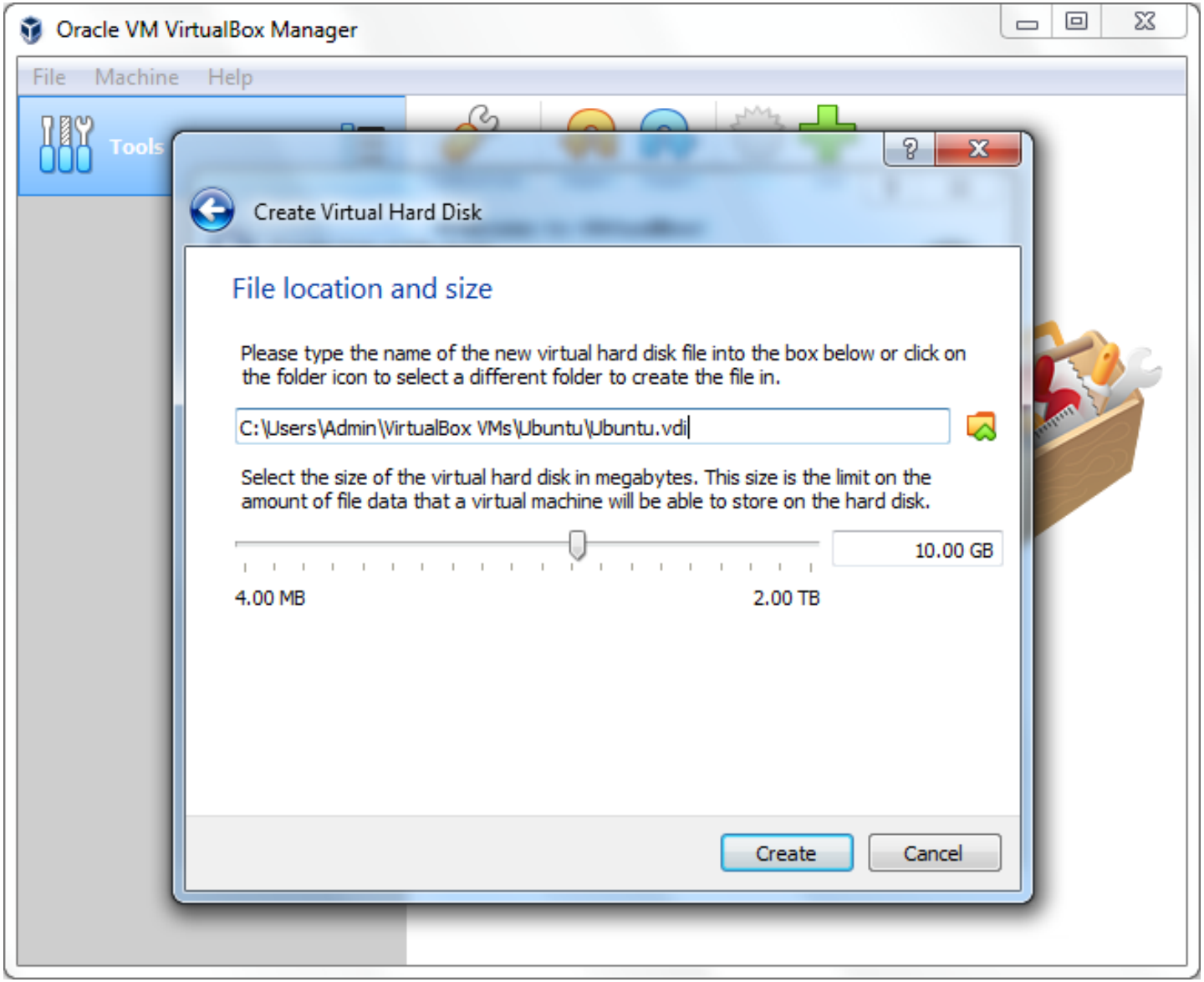
- Select the VDI (Virtual Hard Disk image option) then click next



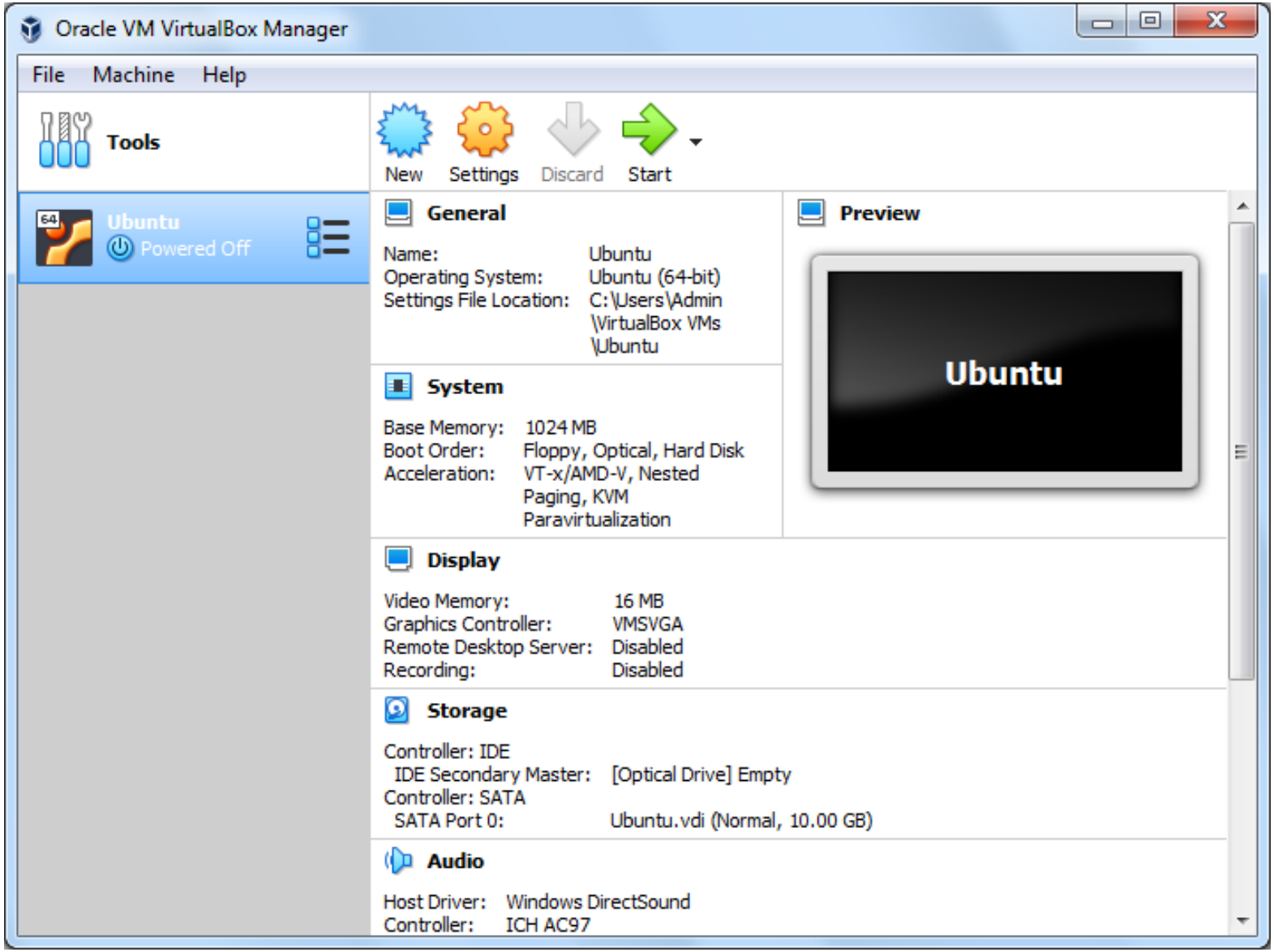
- Select Dynamically allocated option then click next



- Select the size of the virtual hard disk in megabytes the click create button

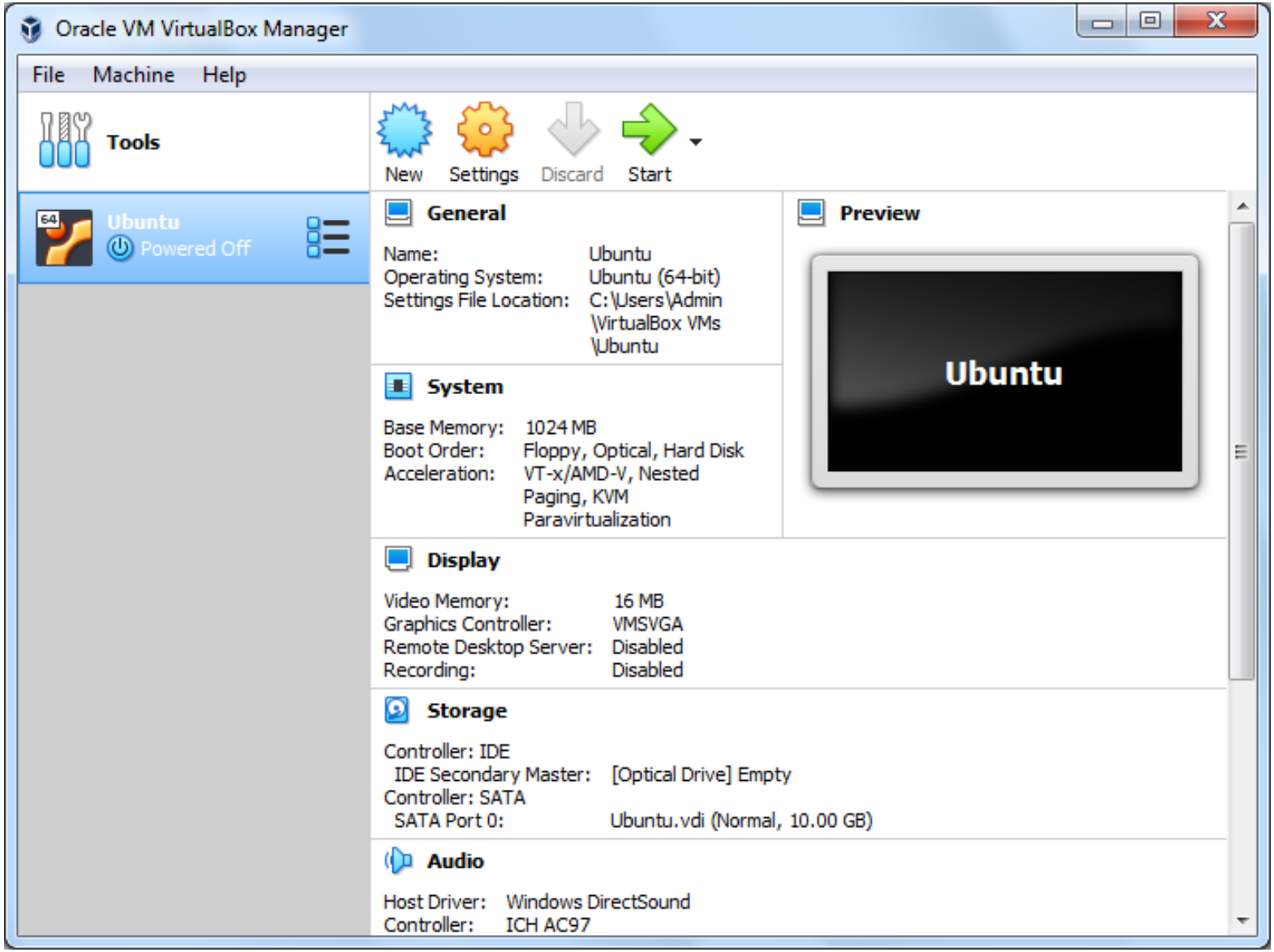


- Created the Virtual Machine Successfully



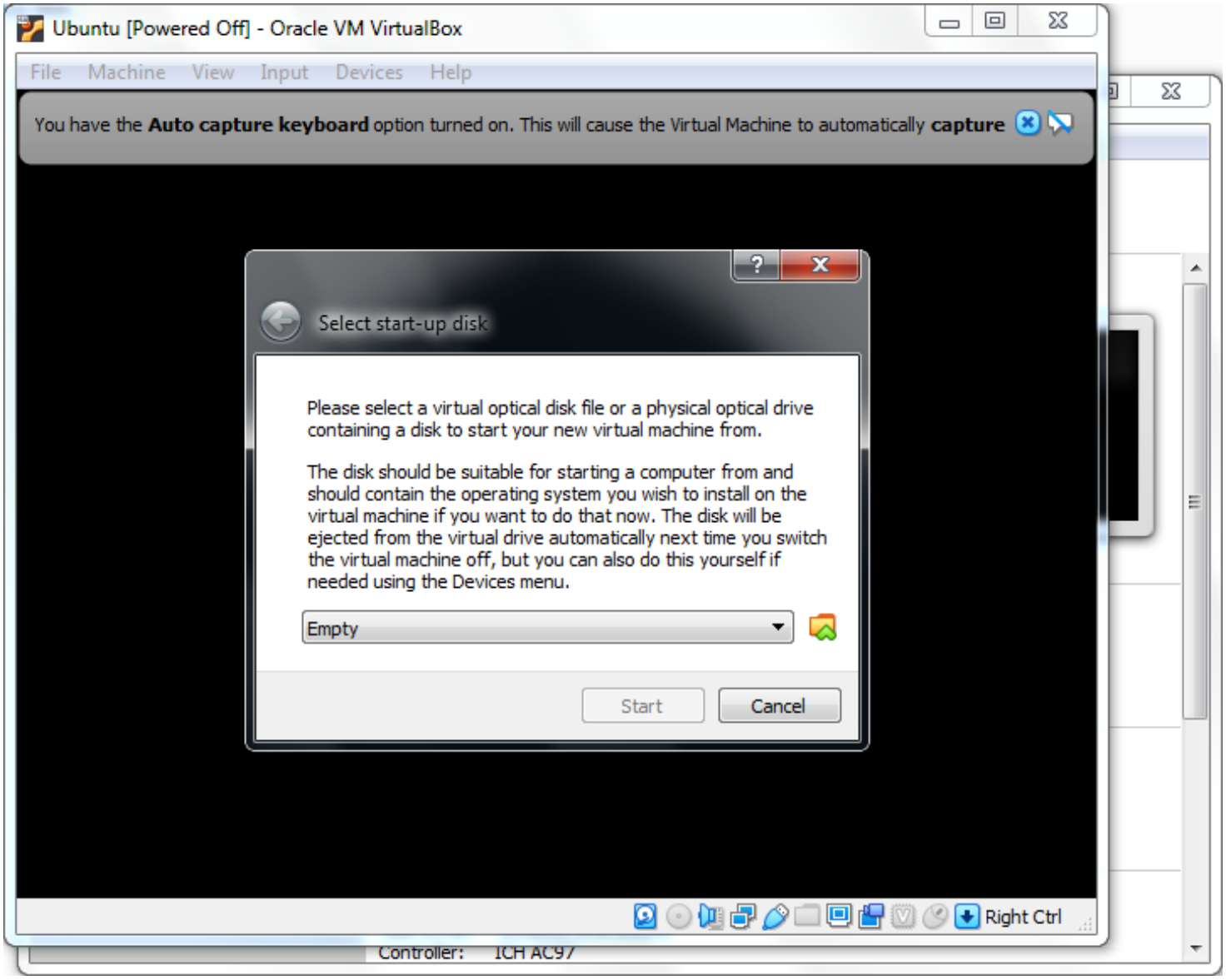
Virtual Box – OS Installation

- Open Oracle Virtual Box Manager

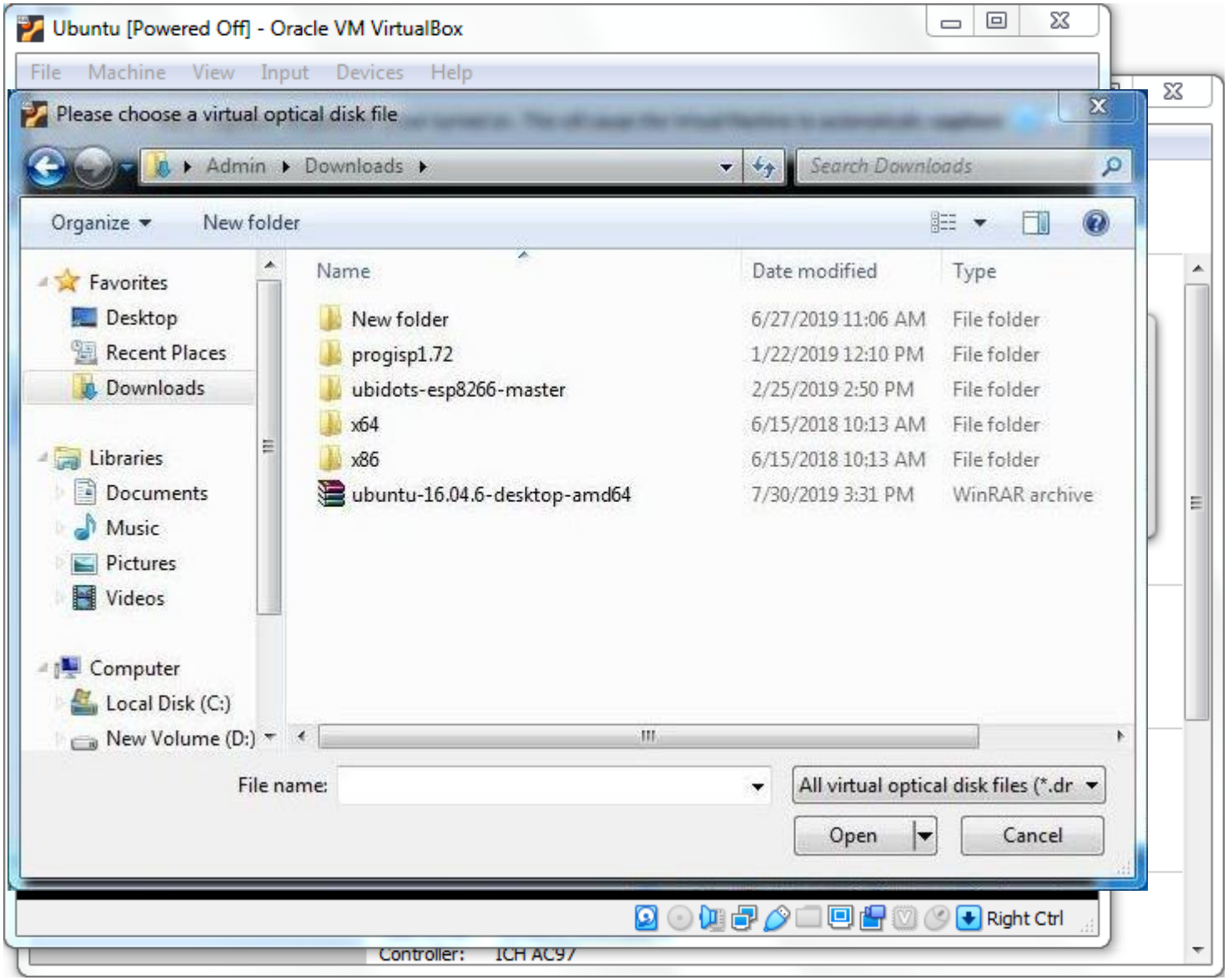


- Click Start to open the Virtual Machine

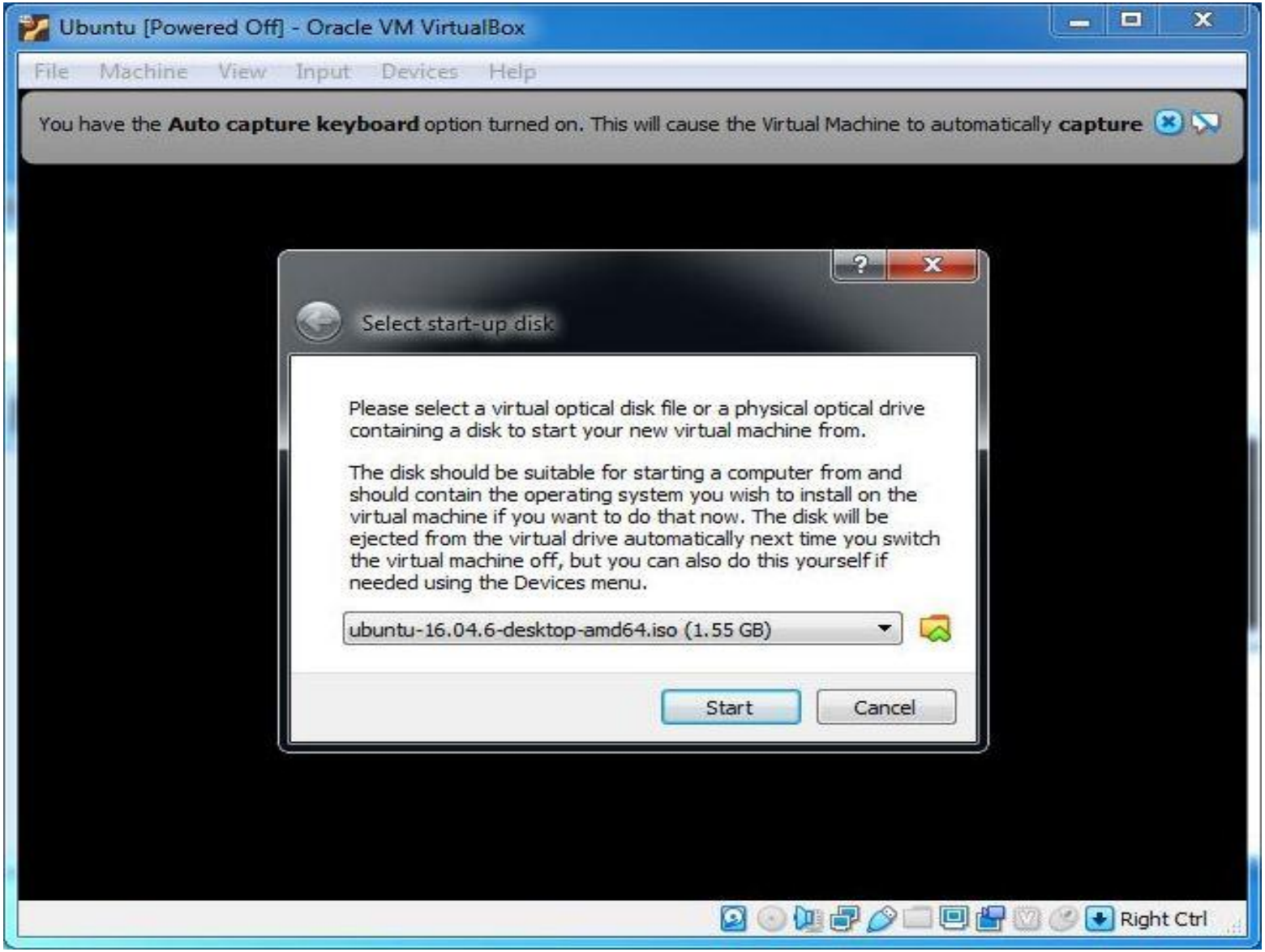
- Click Start the Virtual OS and Select the Startup Image



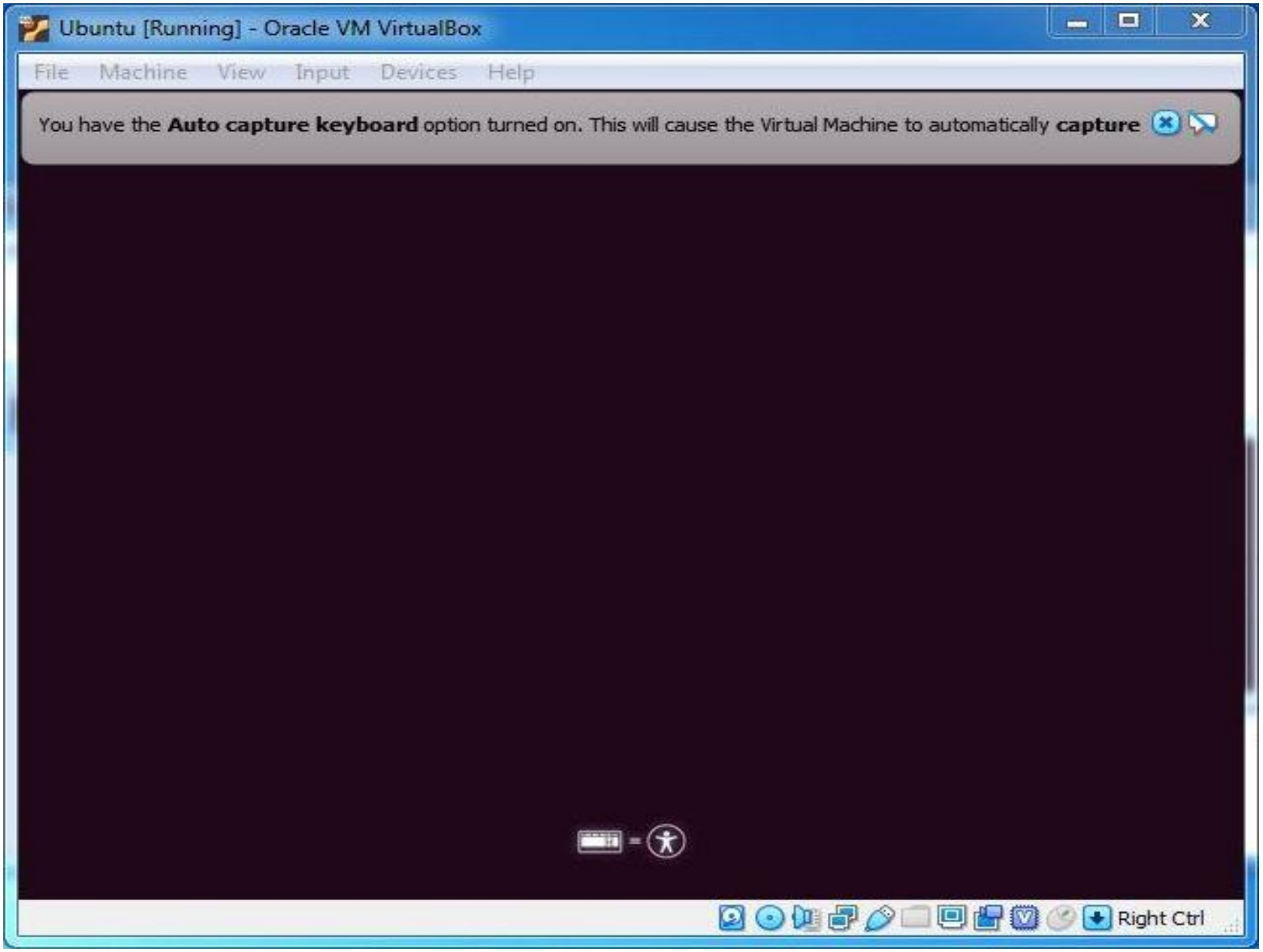
- Select the OS image file then click Open



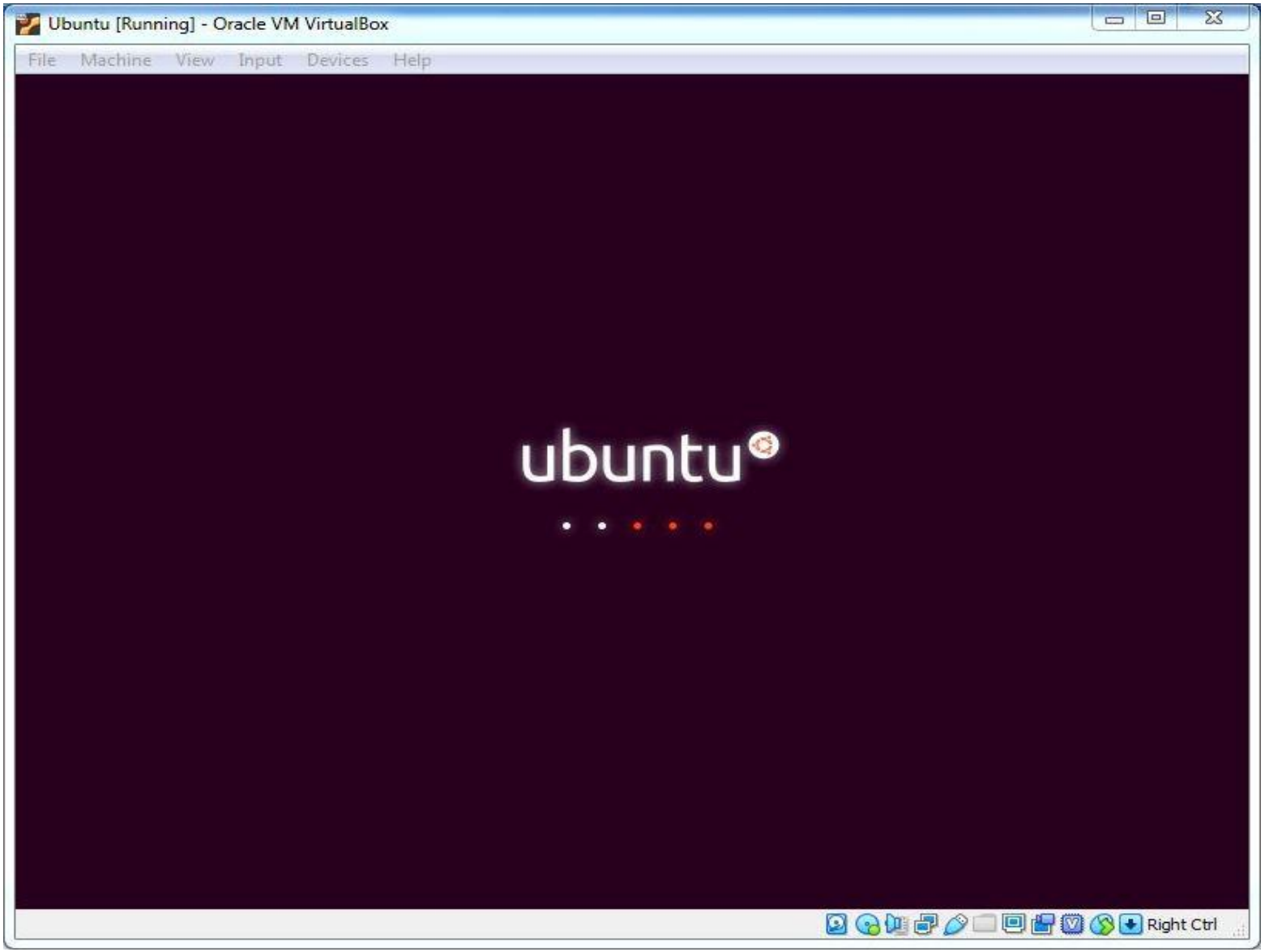
- Startup disk Image file selected click Start



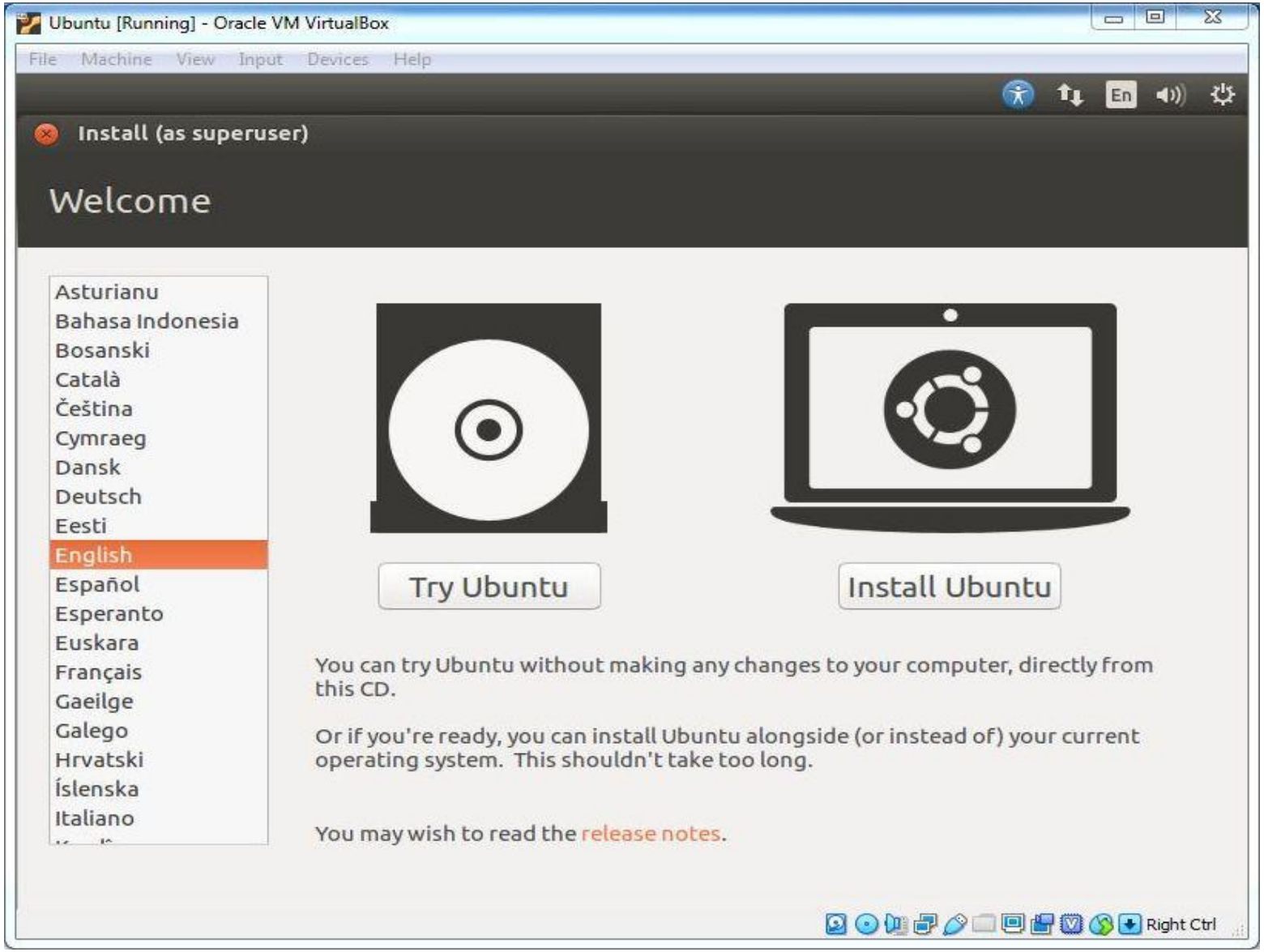
- Virtual Machine Capturing Keyboard, Mouse Automatically



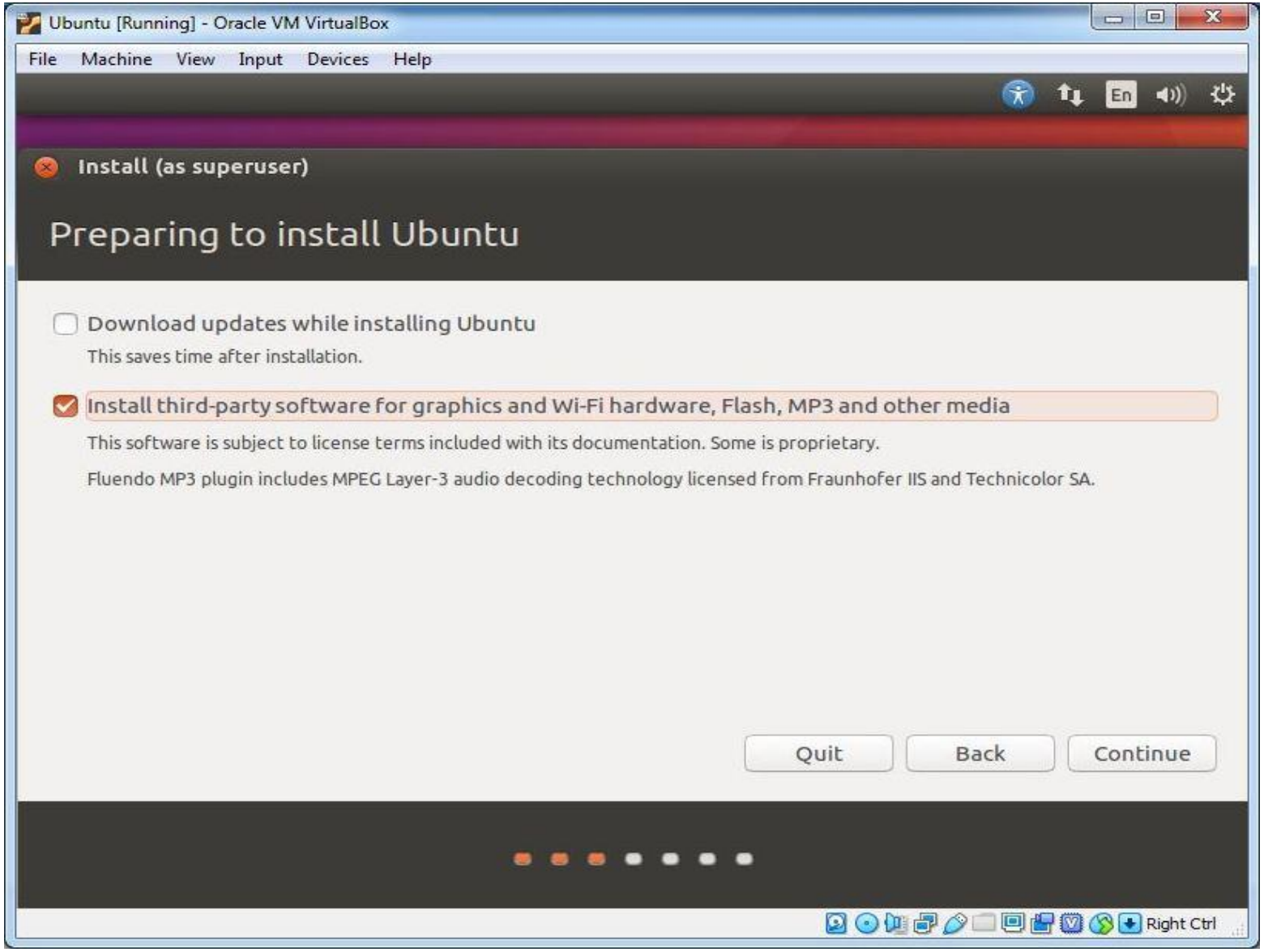
- Virtual Machine Starting to Boot



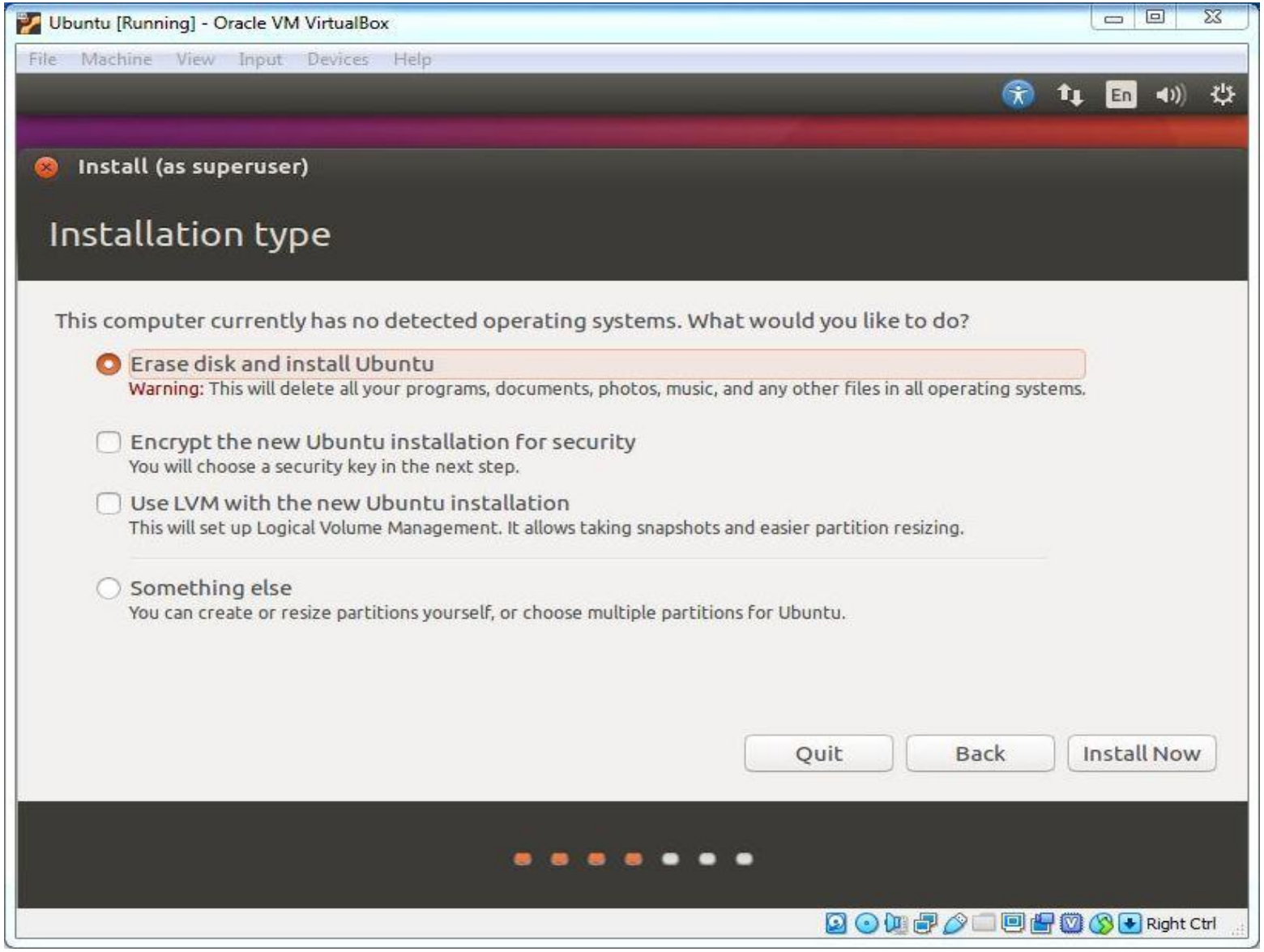
- Click **Install Ubuntu Button**



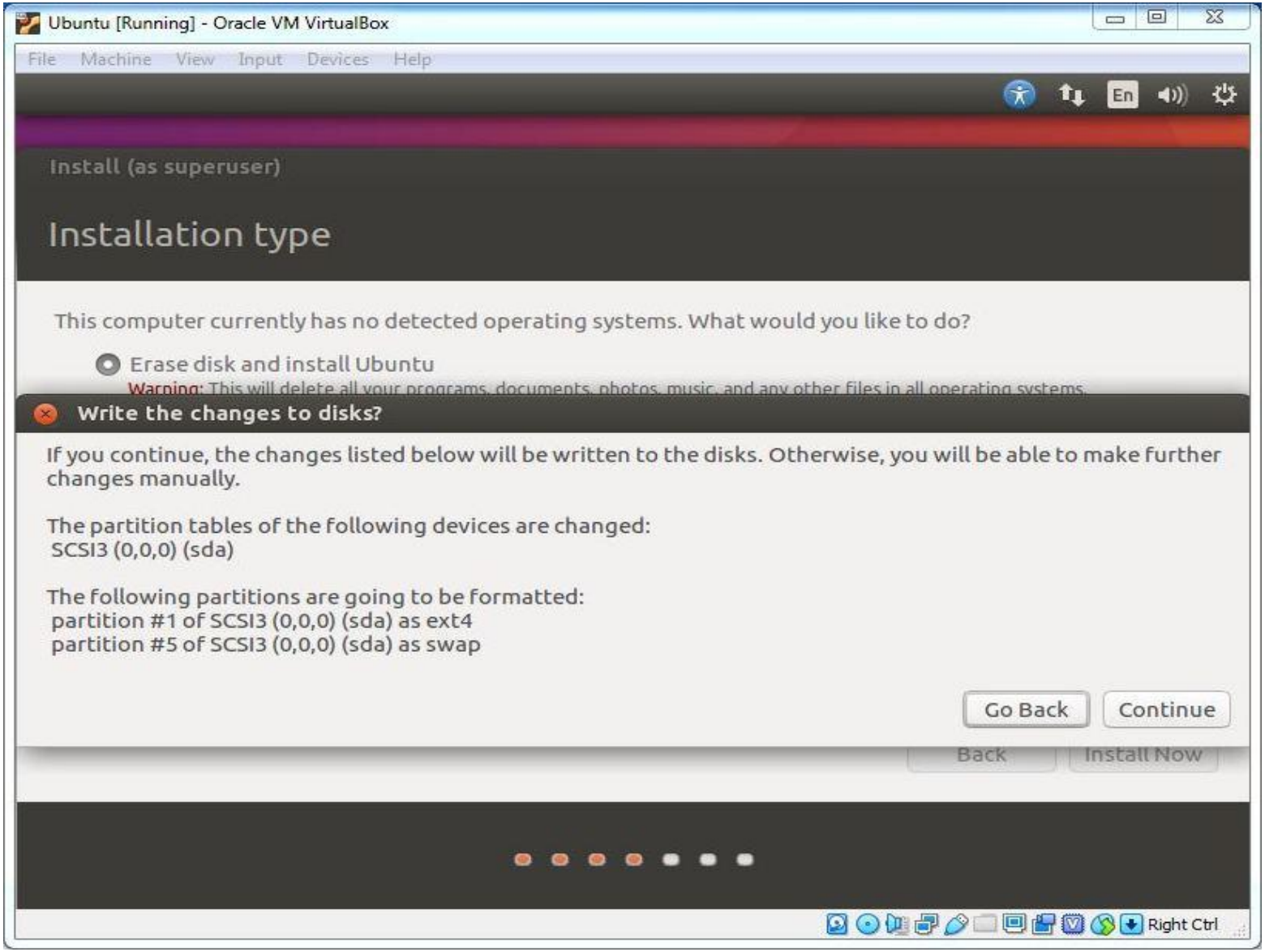
- Select the checkbox install third party software….Click **Continue**



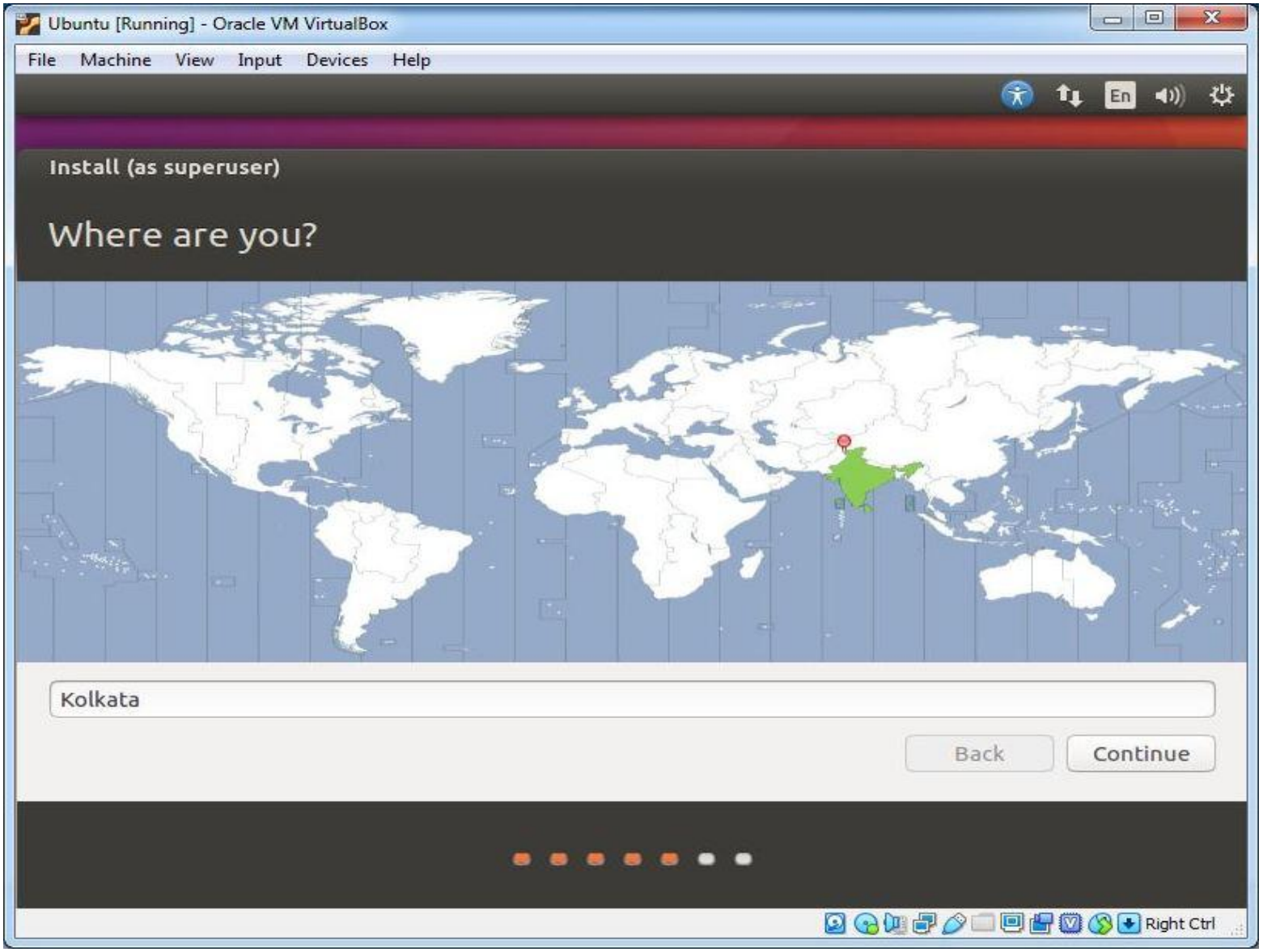
- Select Erase disk and install Ubuntu option click Continue



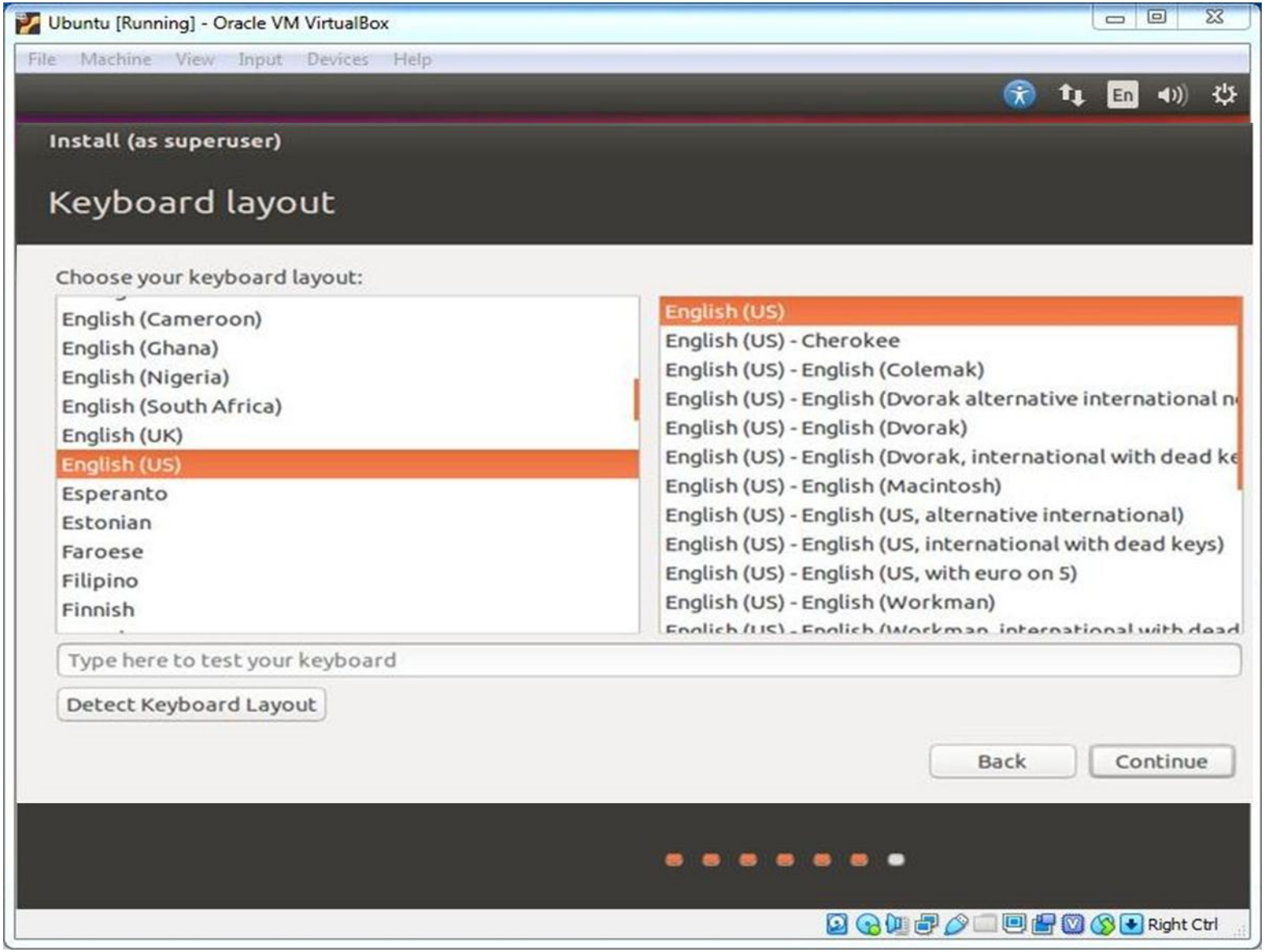
- Click **Continue** to format the Ubuntu partitions



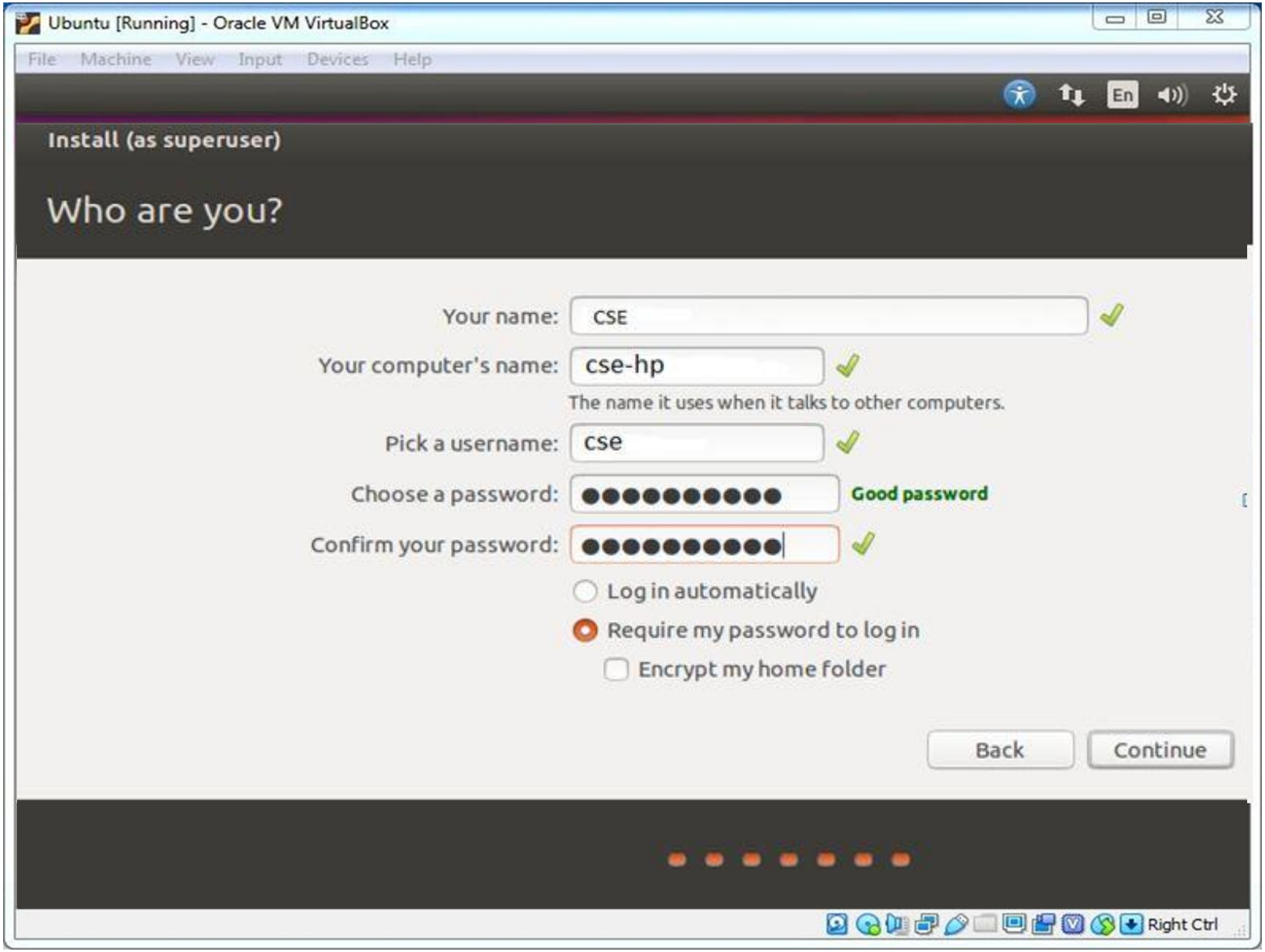
- Select the Country then click **Continue**



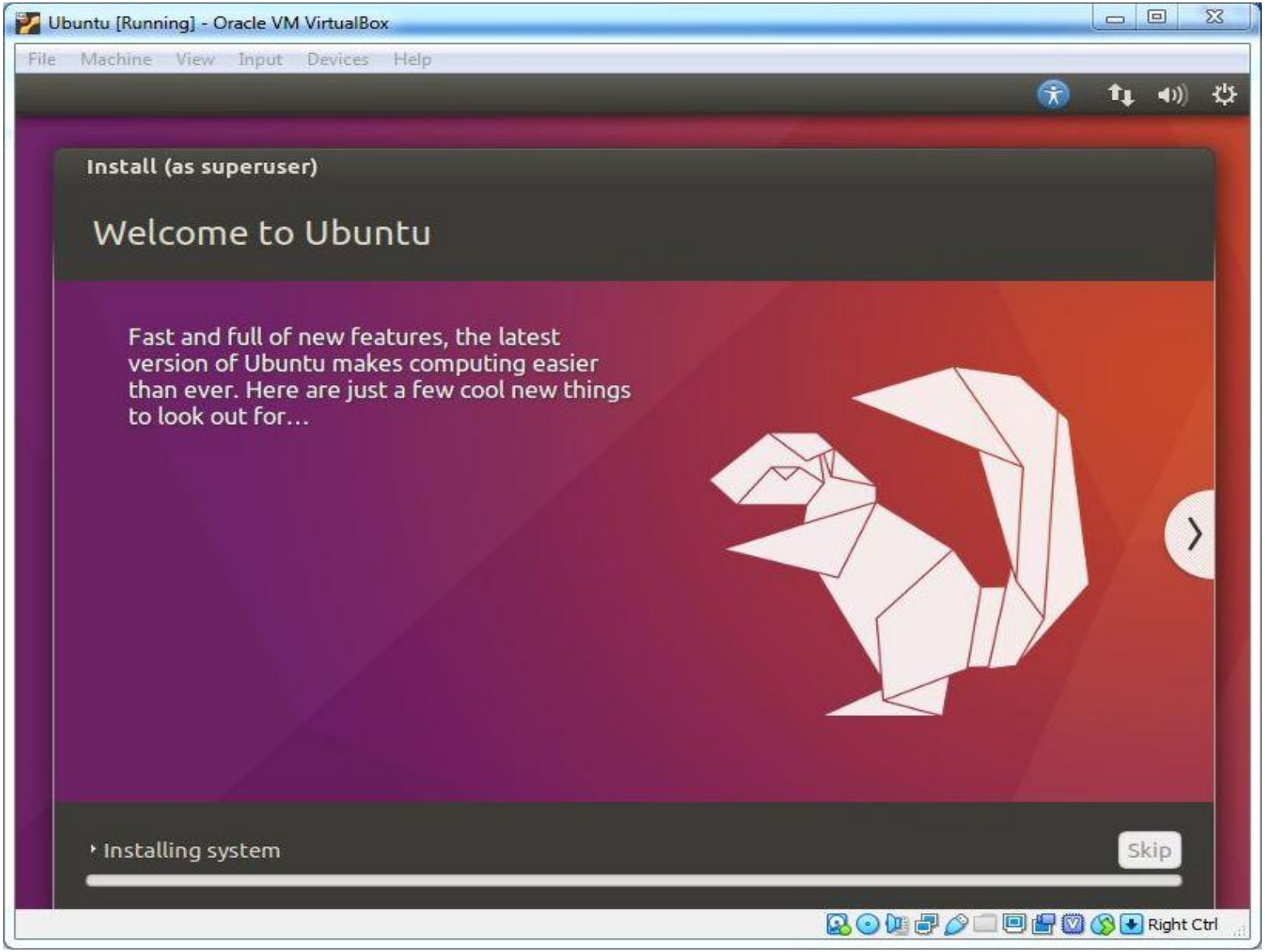
- Select the Keyboard layout click **Continue**



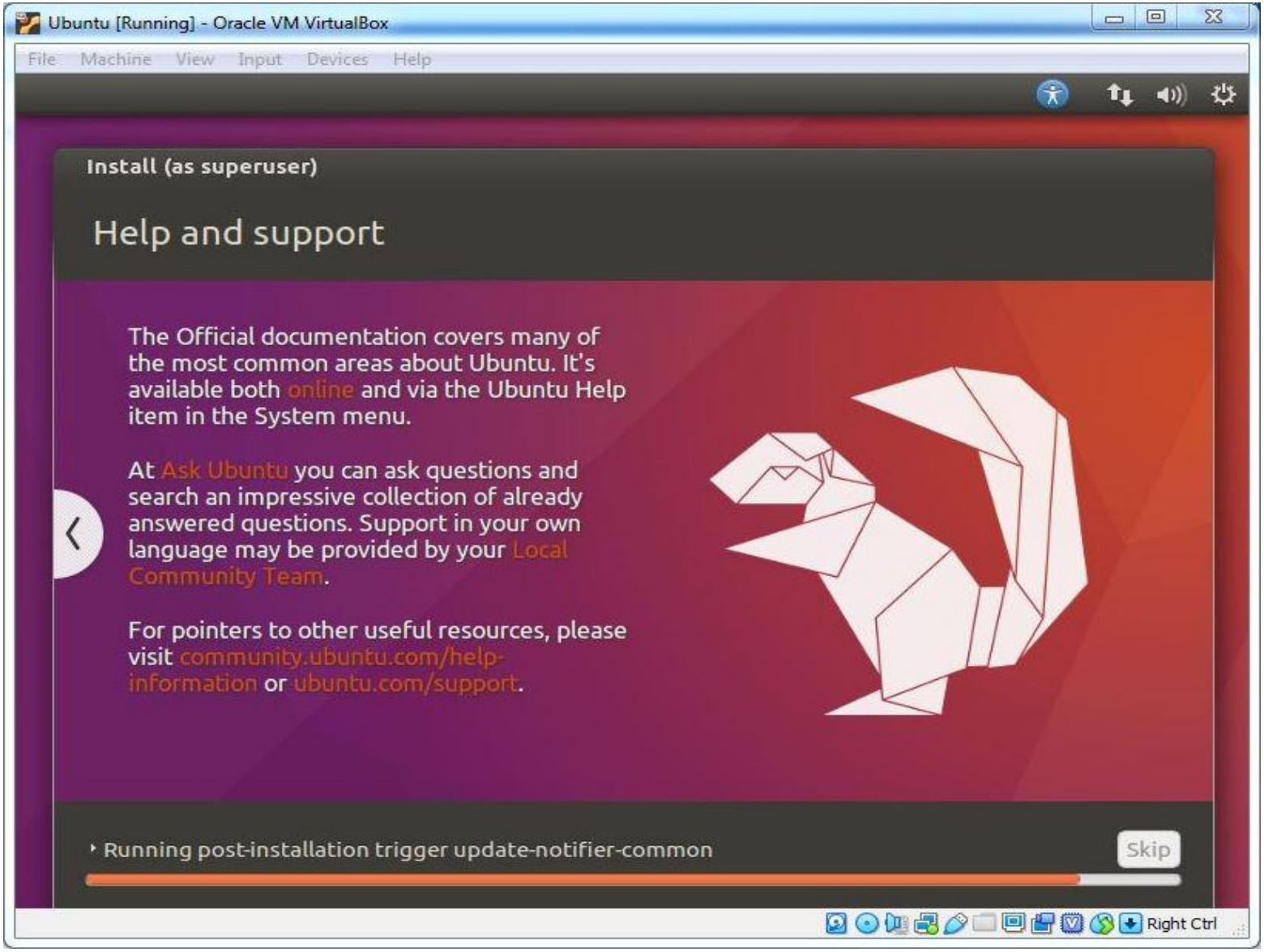
- Put the User Credentials click **Continue**



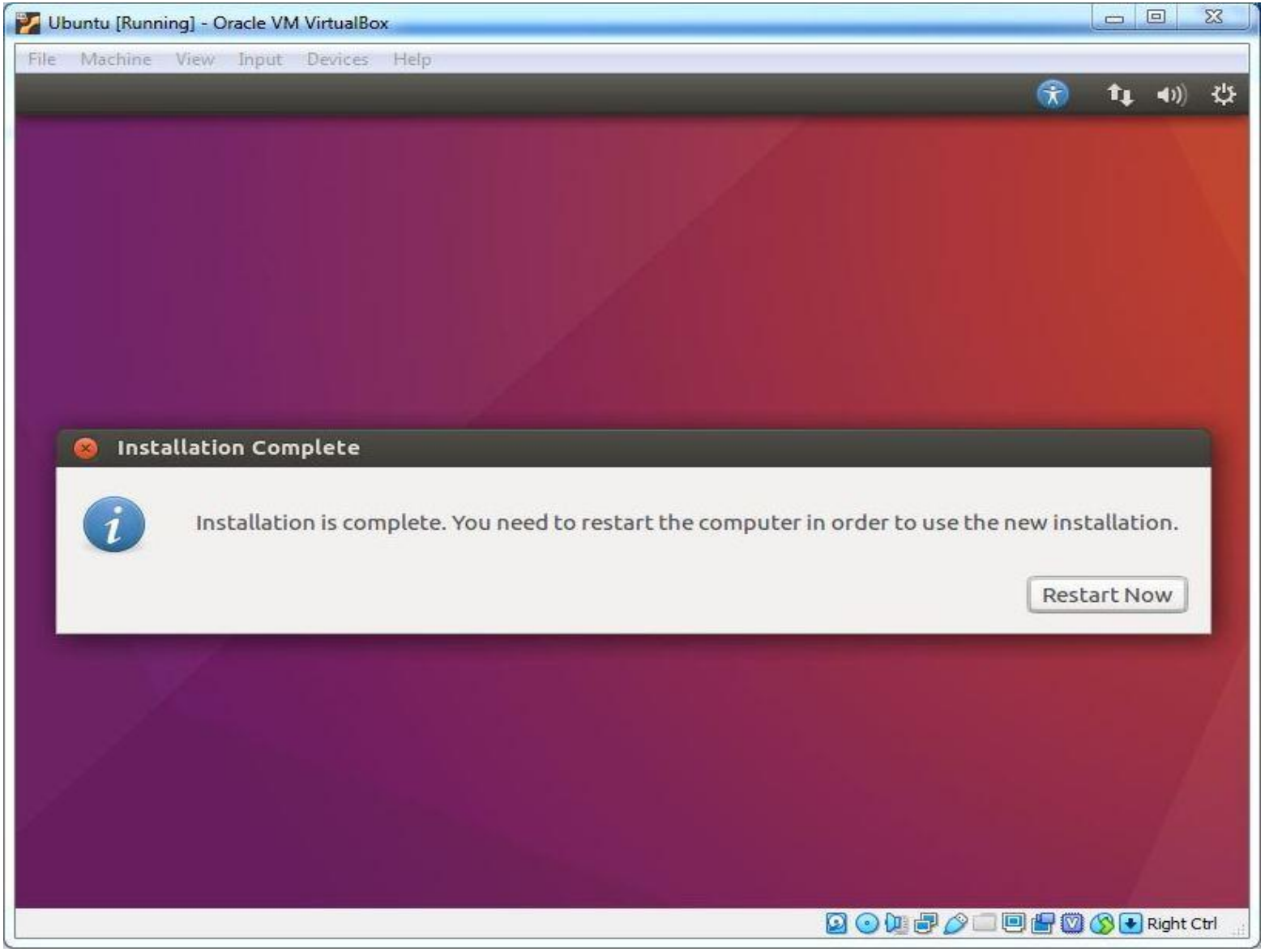
- Installation Begins



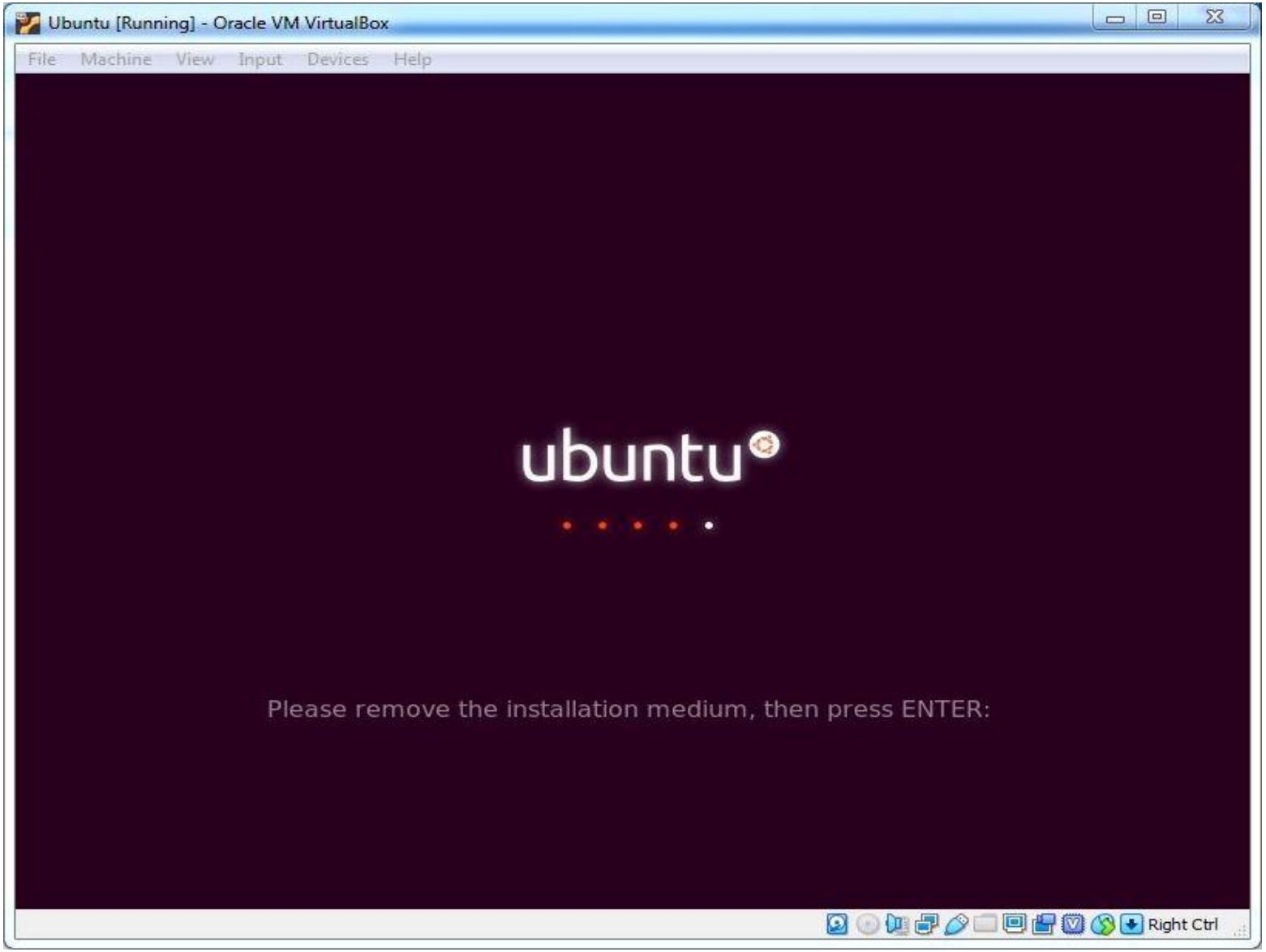
- Almost complete the installation



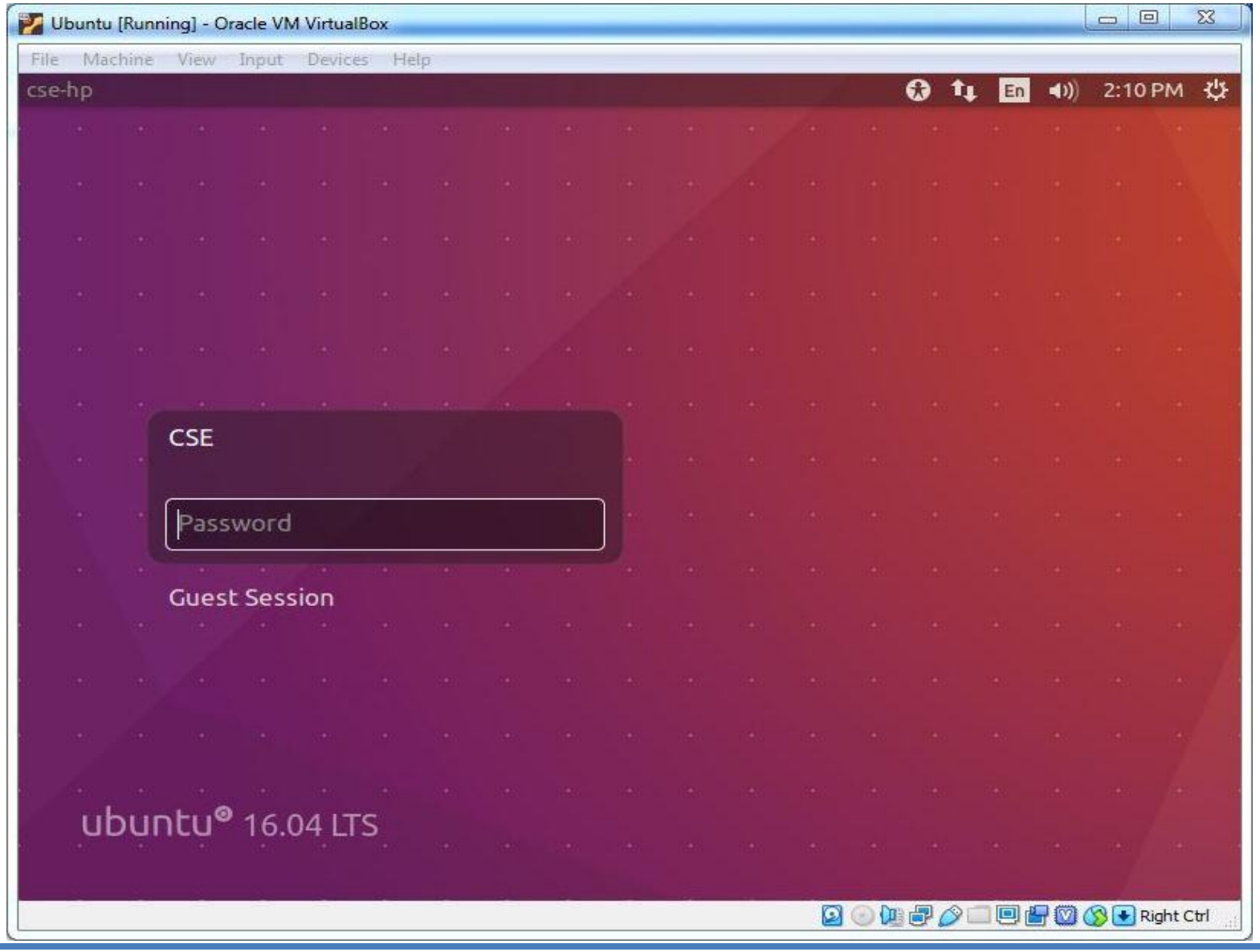
- Installation completed click **Restart Now**



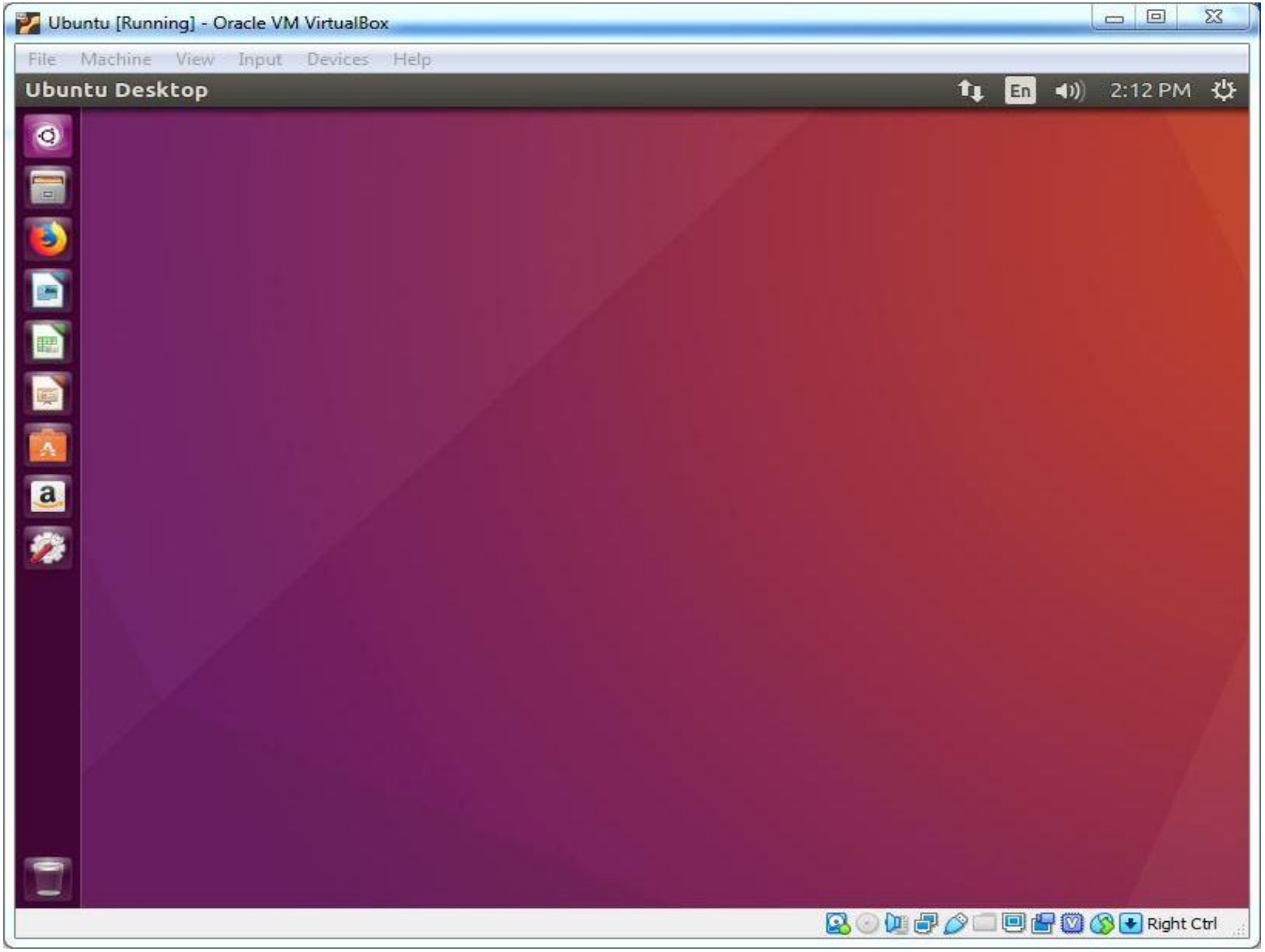
- Press **ENTER key** to restart the machine



- Put the password then press enter to load the OS



- Virtual OS Installed



Add on Cards Installation

Network Interface Card Installation

- Switch off your PC and unplug it from the power connection
- Unplug your monitor, keyboard, mouse and any other peripherals you have connected to your machine.
- Remove the top of your computer case. How you do this depends on the type of *computer chassis* you have. You may need to remove some screws, push two buttons at each side of the case, and/or lift a leaver.
- If you have a *tower chassis*, lay it on its side, motherboard down, then *remove* the side panel.
- Put the screws, or anything else that needs to be re-attached to your PC case, in a safe place so you don't lose them.
- Identify which PCI/PCIe slot you are going to use. If possible, pick one that has the most surrounding space.

- Remove the metal plate for the identified PCI/PCIe slot. A screw may need to be removed. In some PCs, you *push* the metal plate out of position.
- Insert the Ethernet card(NIC). Align it correctly with the PCI/PCIe slot.
- Carefully push the Network Interface Card gently into the PCI/PCIe slot. Make sure both ends of the card are firmly clipped in place. You should be able to see the RJ45 Ports on the outside of your machine.
- Screw the Ethernet card to the computer case and put the computer case back together.
- Reconnect all your peripherals: Keyboard, Mouse, Monitor, Printer, Speakers etc.
- Connect the pc to the power socket and switch on the system.
- After loading the OS, put the driver disc to the optical drive and install the proper driver for the OS.

USB Card Installation

- Switch off your PC and unplug it from the power connection
- Unplug your monitor, keyboard, mouse and any other peripherals you have connected to your machine.
- Remove the top of your computer case. How you do this depends on the type of *computer chassis* you have. You may need to remove some screws, push two buttons at each side of the case, and/or lift a leaver.
- If you have a *tower chassis*, lay it on its side, motherboard down, then *remove* the side panel.
- Put the screws, or anything else that needs to be re-attached to your PC case, in a safe place so you don't lose them.
- Identify which PCI/PCIe slot you are going to use. If possible, pick one that has the most surrounding space.

- Remove the metal plate for the identified PCI slot. A screw may need to be removed. In some PCs, you *push* the metal plate out of position.
- Insert the USB 3.0 card. Align it correctly with the PCI/PCIe slot.
- Carefully push the USB card gently into the PCI/PCIe slot. Make sure both ends of the card are firmly clipped in place. You should be able to see the USB ports on the outside of your machine.
- Screw the USB 3.0 card to the computer case and put the computer case back together.
- Reconnect all your peripherals: Keyboard, Mouse, Monitor, Printer, Speakers etc.
- Connect the pc to the power socket and switch on the system.
- After loading the OS, put the driver disc to the optical drive and install the proper driver for the OS.

Fire Wire Card Installation

- Switch off your PC and unplug it from the power connection
- Unplug your monitor, keyboard, mouse and any other peripherals you have connected to your machine.
- Remove the top of your computer case. How you do this depends on the type of *computer chassis* you have. You may need to remove some screws, push two buttons at each side of the case, and/or lift a leaver.
- If you have a *tower chassis*, lay it on its side, motherboard down, then *remove* the side panel.
- Put the screws, or anything else that needs to be re-attached to your PC case, in a safe place so you don't lose them.
- Identify which PCI/PCIe slot you are going to use. If possible, pick one that has the most surrounding space.

- Remove the metal plate for the identified PCI/PCIe slot. A screw may need to be removed. In some PCs, you *push* the metal plate out of position.

- Insert the Fire Wire card. Align it correctly with the PCI/PCIe slot.

- Carefully push the Fire Wire card gently into the PCI/PCIe slot. Make sure both ends of the card are firmly clipped in place. You should be able to see the Fire Wire (IEEE 1394)ports on the outside of your machine.

- Screw the Fire Wire card to the computer case and put the computer case back together.

- Reconnect all your peripherals: Keyboard, Mouse, Monitor, Printer, Speakers etc.

- Connect the pc to the power socket and switch on the system.

- After loading the OS, put the driver disc to the optical drive and install the proper driver for the OS.

SCSI Card Installation

- Switch off your PC and unplug it from the power connection
- Unplug your monitor, keyboard, mouse and any other peripherals you have connected to your machine.
- Remove the top of your computer case. How you do this depends on the type of *computer chassis* you have. You may need to remove some screws, push two buttons at each side of the case, and/or lift a leaver.
- If you have a *tower chassis*, lay it on its side, motherboard down, then *remove* the side panel.
- Put the screws, or anything else that needs to be re-attached to your PC case, in a safe place so you don't lose them.
- Identify which PCI/PCIe slot you are going to use. If possible, pick one that has the most surrounding space.

- Remove the metal plate for the identified PCI/PCIe slot. A screw may need to be removed. In some PCs, you *push* the metal plate out of position.
- Insert the SCSI card. Align it correctly with the PCI/PCIe slot.
- Carefully push the SCSI card gently into the PCI slot. Make sure both ends of the card are firmly clipped in place. You should be able to see the SCSI card ports on the outside of your machine.
- Screw the SCSI card to the computer case and put the computer case back together.
- Reconnect all your peripherals: Keyboard, Mouse, Monitor, Printer, Speakers etc.
- Connect the pc to the power socket and switch on the system.
- After loading the OS, put the driver disc to the optical drive and install the proper driver for the OS

SCSI Card Installation

- Restart or switch on your computer, and enter your BIOS to disable the integrated sound card.
- Switch off your PC and unplug it from the power connection.
- Unplug your monitor, keyboard, mouse and any other peripherals you have connected to your machine.
- Remove the top of your computer case. How you do this depends on the type of *computer chassis* you have. You may need to remove some screws, push two buttons at each side of the case, and/or lift a leaver.
- If you have a *tower chassis*, lay it on its side, motherboard down, then *remove* the side panel.
- Put the screws, or anything else that needs to be re-attached to your PC case, in a safe place so you don't lose them.
- Identify which PCI/PCIe slot you are going to use. If possible, pick one that has the most surrounding space.

- Remove the metal plate for the identified PCI/PCIe slot. A screw may need to be removed. In some PCs, you *push* the metal plate out of position.
- Insert your new sound. Align it correctly with the PCI/PCIe slot.
- Carefully push or rock the sound card gently into the PCI slot. Make sure both ends of the card are firmly clipped in place. You should be able to see the sound card ports on the outside of your machine.
- Screw the new sound card to the computer case and put the computer case back together.
- Reconnect all your peripherals: Keyboard, Mouse, Monitor, Printer, Speakers etc.
- Connect the pc to the power socket and switch on the system.
- After loading the OS, put the driver disc to the optical drive and install the proper driver for the OS.
- Check the Sound card to play the sample sounds from the PC.

Graphics Card Installation

- Restart or switch on your computer, and enter your BIOS to disable the integrated graphics option
- Switch off your PC and unplug it from the power connection
- Unplug your monitor, keyboard, mouse and any other peripherals you have connected to your machine.
- Remove the top of your computer case. How you do this depends on the type of *computer chassis* you have. You may need to remove some screws, push two buttons at each side of the case, and/or lift a leaver.
- If you have a *tower chassis*, lay it on its side, motherboard down, then *remove* the side panel.
- Put the screws, or anything else that needs to be re-attached to your PC case, in a safe place so you don't lose them.
- Identify which PCI Express slot you are going to use. If possible, pick one that has the most surrounding space.

- Remove the metal plate for the identified PCI Express slot. A screw may need to be removed. In some PCs, you *push* the metal plate out of position.
- Insert your Graphics Card(GPU). Align it correctly with the PCI Express slot.
- Carefully push the Graphics Card gently into the PCI Express slot. Make sure both ends of the card are firmly clipped in place. You should be able to see the Graphics card ports (VGA, DVI, HDMI, Display port) on the outside of your machine.
- Screw the Graphics Card to the computer case and put the computer case back together.
- Reconnect all your peripherals: Keyboard, Mouse, Monitor, Printer, Speakers etc.
- Connect the pc to the power socket and switch on the system.
- After loading the OS, put the driver disc to the optical drive and install the proper driver for the OS.

Operating Systems

Operating System

Operating **S**ystem is a collection of software that manages computer hardware resources and provides common services for computer programs. The operating system is an essential component of the system software in a computer system.

Application programs usually require an operating system to function. Operating System can be defined as “A program that acts as an intermediary between a user of a computer and the computer hardware” Goals of Operating System Are:

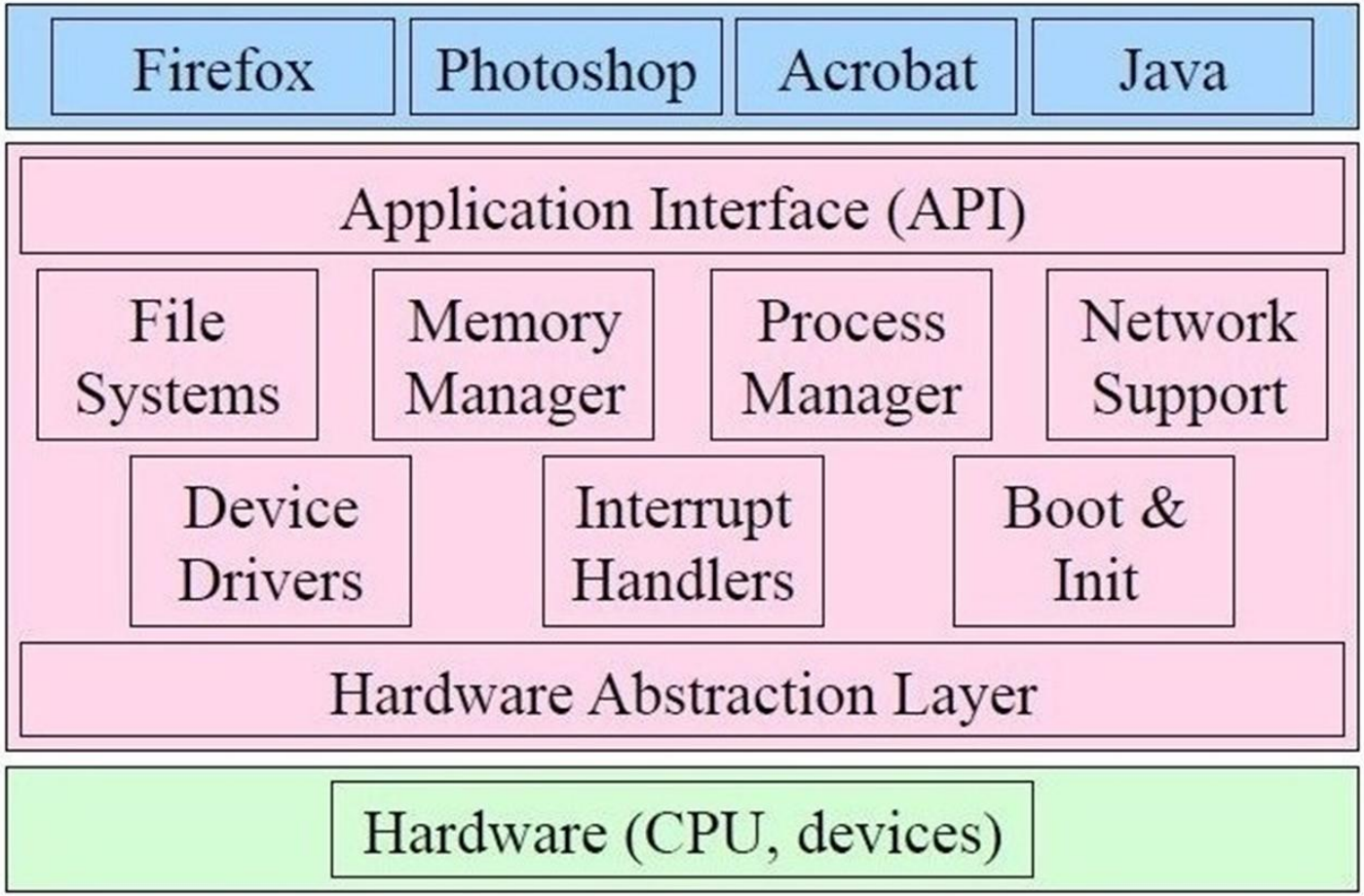
- ❑ Execute user programs and make solving user problems easier
- ❑ Make the computer system convenient to use
- ❑ Use the computer hardware in an efficient manner.

Operating systems can be found on almost any device that contains a computer from cellular phones and video game consoles to supercomputers and web servers.

An operating system is an interface between the hardware of a computer and the user (programs or humans)that facilitates the execution of other programs and the access to hardware and software resources.

Components of Operating System:

The components of an operating system all exist in order to make the different parts of a computer work together. All user software needs to go through the operating system in order to use any of the hardware, whether it be as simple as a mouse or keyboard or as complex as an Internet component.



Functions of an Operating System

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Process Management

A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.

The operating system is responsible for the following activities in connection with process management.

- Process creation and deletion.
- process suspension and resumption.
- Provision of mechanisms for:
  - Process synchronization
  - Process communication

Main-Memory Management

Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.

Main memory is a volatile storage device. It loses its contents in the case of system failure.

The operating system is responsible for the following activities in connections with memory management:

- Keep track of which parts of memory are currently being used and by whom.
- Decide which processes to load when memory space becomes available.
- Allocate and de-allocate memory space as needed.

File Management

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.

The operating system is responsible for the following activities in connections with file management:

- File creation and deletion.
- Directory creation and deletion.
- Support of primitives for manipulating files and directories.
- Mapping files onto secondary storage.
- File backup on stable (nonvolatile) storage media.

I/O System Management

The I/O system consists of:

- A buffer-caching system
- A general device-driver interface
- Drivers for specific hardware devices

Secondary-Storage Management

Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory. Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.

The operating system is responsible for the following activities in connection with disk management:

- Free space management
- Storage allocation
- Disk scheduling

Networking (Distributed Systems)

- A *distributed* system is a collection processors that do not share memory or a clock. Each processor has its own local memory.
- The processors in the system are connected through a communication network.
- Communication takes place using a *protocol.*
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

Protection System

Protection refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.

- The protection mechanism must:
  - Distinguish between authorized and unauthorized usage.
  - Specify the controls to be imposed.
  - Provide a means of enforcement.

Command-Interpreter System

Many commands are given to the operating system by control statements which deal with:

- Process creation and management
- I/O handling
- Secondary-storage management
- Main-memory management
- File-system access
- Protection
- Networking
  - The program that reads and interprets control statements is called variously:
- Command-line interpreter
- Shell (in UNIX)
  - Its function is to get and execute the next command statement.

Operating-System Structures

- System Components
- Operating System Services
- System Calls
- System Programs
- System Structure
- Virtual Machines
- System Design and Implementation
- System Generation

Common System Components

- Process Management
- Main Memory Management
- File Management
- I/O System Management
- Secondary Management
- Networking
- Protection System
- Command-Interpreter System

Operating System Services

User Interface

Means by which users can issue commands to the system. Depending on the system these may be a command-line interface (e.g. sh, csh, ksh, tcsh, etc.), a GUI interface (e.g. Windows, X-Windows, KDE, Gnome, etc.), or a batch command systems. The latter are generally older systems using punch cards of job-control language, JCL, but may still be used today for specialty systems designed for a single purpose.

Program Execution

The purpose of computer systems is to allow the user to execute programs. So the operating system provides an environment where the user can conveniently run programs. Running a program involves the allocating and de-allocating memory, CPU scheduling in case of multiprocessing

I/O Operations

Each program requires an input and produces output. This involves the use of I/O. So the operating systems are providing I/O makes it convenient for the users to run programs.

File System Manipulation

The output of a program may need to be written into new files or input taken from some files. The operating system provides this service.

Communications

The processes need to communicate with each other to exchange information during execution. It may be between processes running on the same computer or running on the different computers. Communications can be occur in two ways: (i) shared memory or (ii) message passing

Error Detection

An error is one part of the system may cause malfunctioning of the complete system. To avoid such a situation operating system constantly monitors the system for detecting the errors. This relieves the user of the worry of errors propagating to various part of the system and causing malfunctioning.

Types of User Interface

- Command line interface
- Graphical user interface

Command Line Interface (CLI)

- A**command-ineinterface** is a mechanism for interacting with a computer operating system or software by typing commands to perform specific tasks.
- Thismethodofinstructingacomputertoperformagiventaskisreferredtoas"entering" acommand.
- Acceptinputviakeyboardonly.
- Notsuitableforbeginners.

Graphical User Interface (GUI)

- Is a type of user interface which allows people to interact with computer with images rather than text commands.
- Accept input via keyboard and pointing devices.
- Easy to learn.

Elements of Graphical User Interface

- Pointer
- Icons
- Desktop
- Windows
- Menus

Pointer

- A symbol that appears on the display screen and that you move to select objects and commands.
- Usually, the pointer appears as a small angled arrow.

Icons

- Small pictures that represent commands , files , or windows.

Desktop

- The area on the display screen where icons are grouped is often referred to as the desktop because the icons are intended to represent real objects on a real desktop.

Windows

- Used to divide the screen into different areas.
- In each window , you can run a different program or display a different file.

Menus

- Most graphical user interfaces let you execute commands by selecting a choice from a menu.
- Two types of menu:
  - Pull-down menu
  - Pop-up menu

Types of User Interface

	CLI	GUI
Ease	Because of the memorization and familiarity needed to operate a command line interface, new users have a difficult time navigating and operating a command line interface.	Although new users may have a difficult time learning to use the mouse and all GUI features, most users pick up this interface much easier when compared to a command line interface.
Control	Users have much more control of their file system and operating system in a command line interface. For example, users can copy a specific file from one location to another with a one-line command.	Although a GUI offers plenty of control of a file system and operating system, the more advanced tasks may still need a command line.

	CLI	GUI
Multitasking	Although many command line environments are capable of multitasking, they do not offer the same ease and ability to view multiple things at once on one screen.	GUI users have windows that enable a user to view, control, and manipulate multiple things at once and is much faster to navigate when compared with a command line.
Speed	Command line users only need to use their keyboards to navigate a command line interface and often only need to execute a few lines to perform a task.	A GUI may be easier to use because of the mouse. However, using a mouse and keyboard to navigate and control your operating system for many things is going to be much slower than someone who is working in a command line.

Examples of OS

Microsoft DOS

- DOS stands for **Disk Operating System**.
- Developed by Microsoft Inc.
- Using command line interface.
- It does not support multiple users and multitasking.

Windows

- Produced by Microsoft Inc.
- Using graphical user interface.
- Support multitasking and multiuser

Mac OS

- The official name of the Macintosh operating system.
- Created by Apple Inc.
- Operating System for Apple Macintosh computer.

Examples of OS Cont..,

Linux

- A freely- distributable open source operating system that runs on a number of hardware platforms.
- Linux has become an extremely popular alternative to proprietary operating systems.

Solaris

- **Solaris** is a Unix operating system originally developed by Sun Microsystems
- Solaris can be installed from physical media or a network for use on a desktop or server
- Used on server and workstation

Android

- **Android** is a Linux-based operating system for mobile devices such as smartphones and tablet computers
- •It is developed by the Open Handset Alliance, led by Google, and other companies

Efficient Operation of the OS

Resource Allocation

- E.g. CPU cycles, main memory, storage space, and peripheral devices. Some resources are managed with generic systems and others with very carefully designed and specially tuned systems, customized for a particular resource and operating environment.

Accounting

- Keeping track of system activity and resource usage, either for billing purposes or for statistical record keeping that can be used to optimize future performance.

Protection and Security

- Preventing harm to the system and to resources, either through wayward internal processes or malicious outsiders. Authentication, ownership, and restricted access are obvious parts of this system. Highly secure systems may log all process activity down to excruciating detail, and security regulation dictate the storage of those records on permanent non-erasable medium for extended times in secure (off-site) facilities.

System calls

- Provide a means for user or application programs to call upon the services of the operating system. Generally written in C or C++, although some are written in assembly for optimal performance.

OS Evolution

- Operating systems have gone through a long history of evolution, which we summarize here.
  - Batch systems
  - Time-sharing systems
  - Personal systems
  - Parallel systems
  - Distributed systems
  - Real-time systems

Batch systems

- **Batch operating systems** were designed in the 1950s to control mainframe computers. At that time, computers were large machines that used punched cards for input, line printers for output and tape drives for secondary storage media.

- Each program to be executed was called a job. **A programmer who wished to execute a job sends a request to the operating system**.

Time-sharing systems

- To use computer system resources efficiently, *multiprogramming* was introduced. The idea is to hold several jobs in memory at a time, and only assign a resource to a job that needs it on the condition that the resource is available.

- Multiprogramming brought the idea of **time sharing**: **resources could be shared between different jobs, with each job being allocated a portion of time to use a resource**. Because a computer is much faster than a human, time sharing is hidden from the user—each user has the impression that the whole system is serving them exclusively.

Personal systems

- When personal computers were introduced, there was a need for an operating system for this new type of computer.
- During this era, **single-user operating systems** such as **DOS (Disk Operating System)** were introduced.

Parallel systems

- The need for more speed and efficiency led to the design of **parallel systems**: **multiple CPUs on the same machine**. Each CPU can be used to serve one program or a part of a program, which means that many tasks can be accomplished in parallel instead of serially.
- The operating systems required for this are more complex than those that support single CPUs.

Distributed systems

- Networking and internetworking have created a new dimension in operating systems. A job that was previously done on one computer can now **be shared between computers that may be thousands of miles apart**.
- Distributed systems combine features of the previous generation with new duties such as controlling security.

Real-time systems

- **A real-time system is expected to do a task within a specific time constraint**. They are used with real-time applications, which monitor, respond to or control external processes or environments.

Types of Operating System

- Real-time Operating System
- Multi-user Operating System
- Multi-tasking Operating System
- Distributed Operating System
- Embedded Operating System
- Time Sharing Operating System

Real-time Operating System

- A real-time operating system is a multitasking operating system that aims at executing real-time applications. Real-time operating systems often use specialized scheduling algorithms so that they can achieve a deterministic nature of behavior.

- The main objective of real-time operating systems is their quick and predictable response to events. They have an event-driven or time-sharing design and often aspects of both.

- An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts.

Multi-user Operating System

- A multi-user operating system allows multiple users to access a computer system at the same time. Time-sharing systems and Internet servers can be classified as multi-user systems as they enable multiple-user access to a computer through the sharing of time.

- Single-user operating systems have only one user but may allow multiple programs to run at the same time.

Multi-tasking Operating System

- A multi-tasking operating system allows more than one program to be running at the same time, from the point of view of human time scales. A single-tasking system has only one running program.

- Multi-tasking can be of two types: pre-emptive and co-operative.

  - In pre-emptive multitasking, the operating system slices the CPU time and dedicates one slot to each of the programs. Unix-like operating systems such as Solaris and Linux support pre-emptive multitasking, as does AmigaOS.

  - Cooperative multitasking is achieved by relying on each process to give time to the other processes in a defined manner. 16-bit versions of Microsoft Windows used cooperative multi-tasking. 32-bit versions of both Windows NT and Win9x used pre-emptive multi-tasking. Mac OS prior to OS X used to support cooperative multitasking.

Distributed Operating System

- A distributed operating system manages a group of independent computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they make a distributed system.

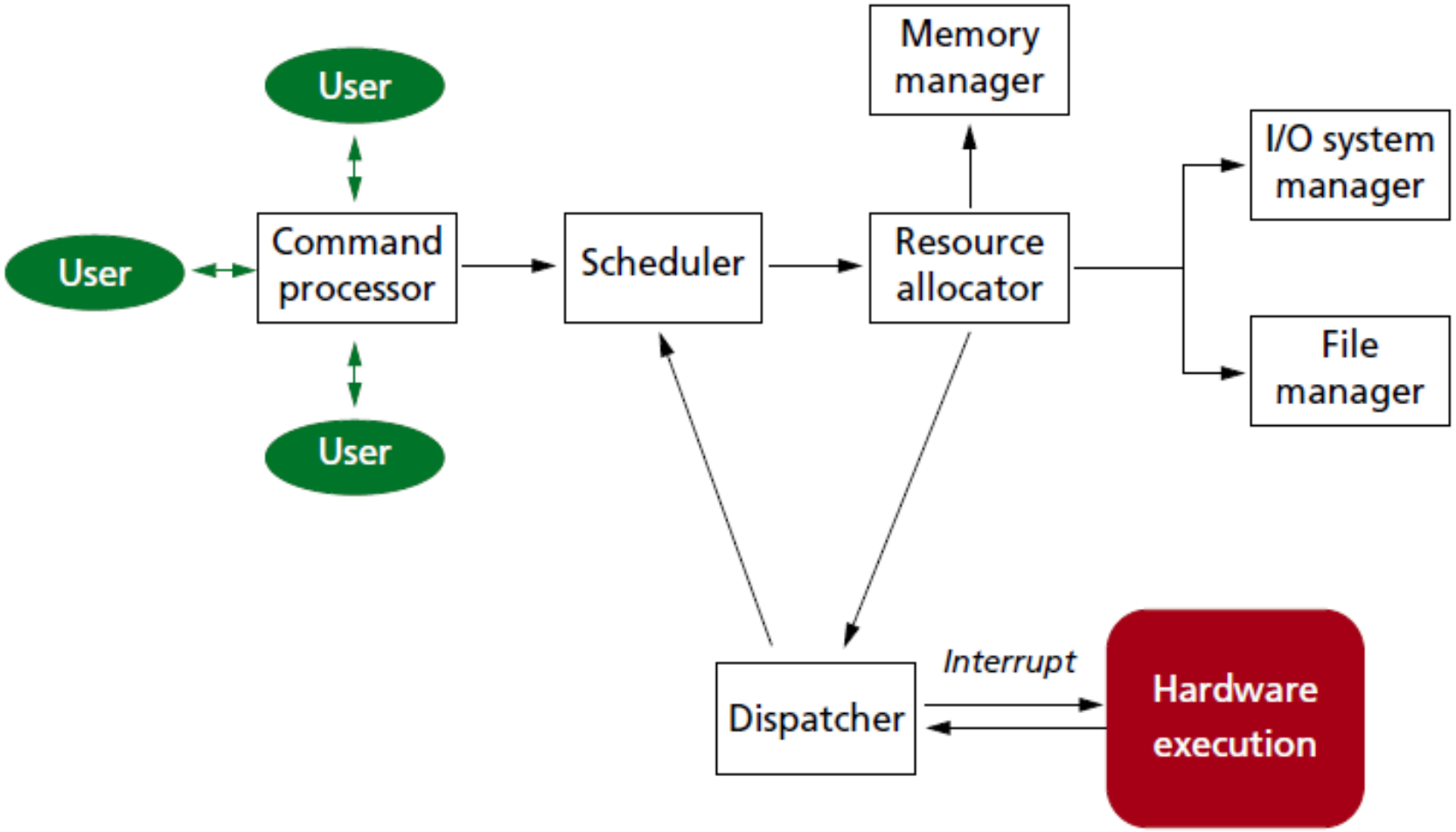
Embedded Operating System

- Embedded operating systems are designed to be used in embedded computer systems. They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design.

Time Sharing Operating System

- Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.
- Examples of popular modern operating systems include Android, BSD, iOS, Linux, OS X, QNX, Microsoft Windows, Windows Phone, and IBM z/OS. All these, except Windows, Windows Phone and z/OS, share roots in UNIX.

Operating System Architecture



Command processor:

- The interface that interacts with one or more users, watching the keyboard, mouse, and any other input devices attached to the machine.
- Receives commands whenever an input device notifies it about an **event** , for example, when a user enters a new line in a shell or clicks on a mouse button.

Scheduler:

- If the event turns out to be a **request to run a program** , the command processor asks the scheduler to arrange for the execution of programs.
  - Place the program in a job queue.
  - Create a process to execute the program.
- A **process** is an active copy of a program
  - That has been loaded into memory (RAM)
  - With its own program counter indicating the next instruction to execute.
- Each process creates its own **Context** (= process state).

  There may be more than one process for the same program (e.g., start two different editor programs).

Resource Allocator

- The scheduler invokes a resource allocator, which makes sure that each process has **secondary resources** that it may need — memory, files, peripheral devices.

Memory Manager

The memory manager coordinates the use of the machine's memory.

- It keeps track of which areas of memory are being used by which processes.
- It may anticipate how much memory each process will need.
- It may provide virtual memory, where "swap files" on a hard disk drive are used to extend the RAM.

I/O System Manager

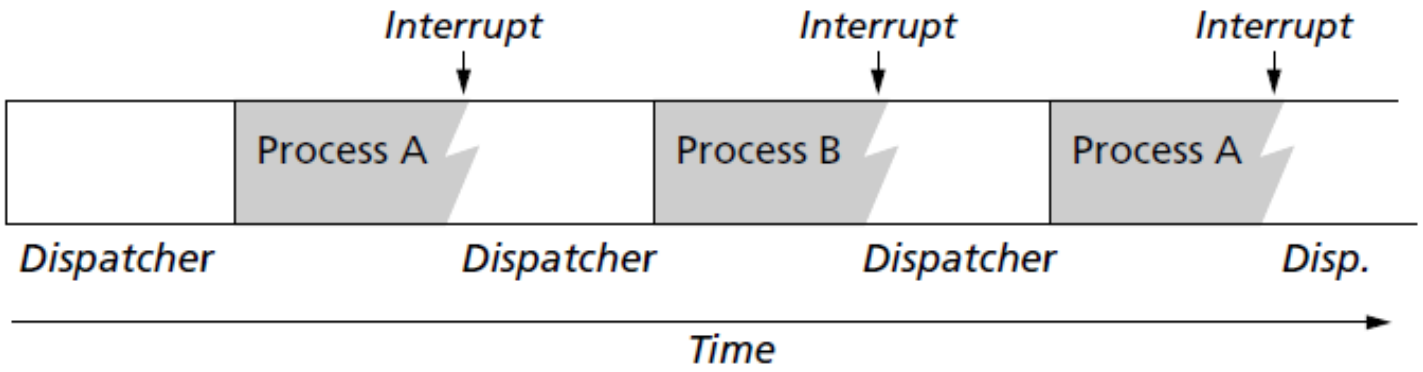
- The I/O system manager coordinates the assignment of peripheral devices to processes, through specific device drivers.

File Manager

- The file manager keeps track of information about files on the disks:
  - Protects against unauthorized file access
  - Supports sharing of file resources
  - Helps organizing files into folders

Dispatcher

- The dispatcher monitors processes and decides when to switch execution from one process to another.



When a process completes ...

- The dispatcher reports back to the scheduler,
- The scheduler notifies the resource allocator,
- The resource allocator releases any resources held by that process,
- The scheduler then reports completion of the process to the command processor,
- The command processor informs the user.

Difference between 32-Bit vs. 64 Bit Operating System

Parameters	32Bit	64 Bit
Architecture and Software	Allow 32 bit of data processing simultaneously	Allow 64 bit of data processing simultaneously
Compatibility	32-bit applications require 32-bit OS and CPUs.	64-bit applications require a 64-bit OS and CPU.
Systems Available	All versions of Windows 8, Windows 7, Windows Vista, and Windows XP, Linux, etc.	Windows XP Professional, Vista, 7, Mac OS X and Linux.
Memory Limits	32-bit systems are limited to 3.2 GB of RAM.	64-bit systems allow a maximum 17 Billion GB of RAM.

Virtual Machine

Virtual Machine

- A virtual machine is a computer file, typically called an image, which behaves like an actual computer. In other words, creating a computer within a computer.
- It runs in a window, much like any other programme, giving the end user the same experience on a virtual machine as they would have on the host operating system itself.
- The virtual machine is sandboxed from the rest of the system, meaning that the software inside a virtual machine cannot escape or tamper with the computer itself.
- This produces an ideal environment for testing other operating systems including beta releases, accessing virus-infected data, creating operating system backups and running software or applications on operating systems for which they were not originally intended.
- Multiple virtual machines can run simultaneously on the same physical computer. For servers, the multiple operating systems run side-by-side with a piece of software called a hypervisor to manage them, while desktop computers typically employ one operating system to run the other operating systems within its programme windows

- Each virtual machine provides its own virtual hardware, including CPUs, memory, hard drives, network interfaces and other devices.
- The virtual hardware is then mapped to the real hardware on the physical machine which saves costs by reducing the need for physical hardware systems along with the associated maintenance costs that go with it, plus reduces power and cooling demand.
- A virtual machine is a program that acts as a virtual computer. It runs on your current operating system (the host operating system) and provides virtual hardware to guest operating systems. The guest OS runs in a window on your host OS, just like any other program on your computer.
- You can have several virtual machines installed on your system. You're only limited by the amount of storage you have available for them.
- Once you've installed several operating systems, you can open your virtual machine program and choose which virtual machine you want to boot.
- The guest operating system starts up and runs in a window on your host operating system, although you can also run it in full-screen mode.

- A VM is very useful in organizations for many reasons, from application development and testing in different environments to data backup and storage, etc. as it proves to be cost effective.

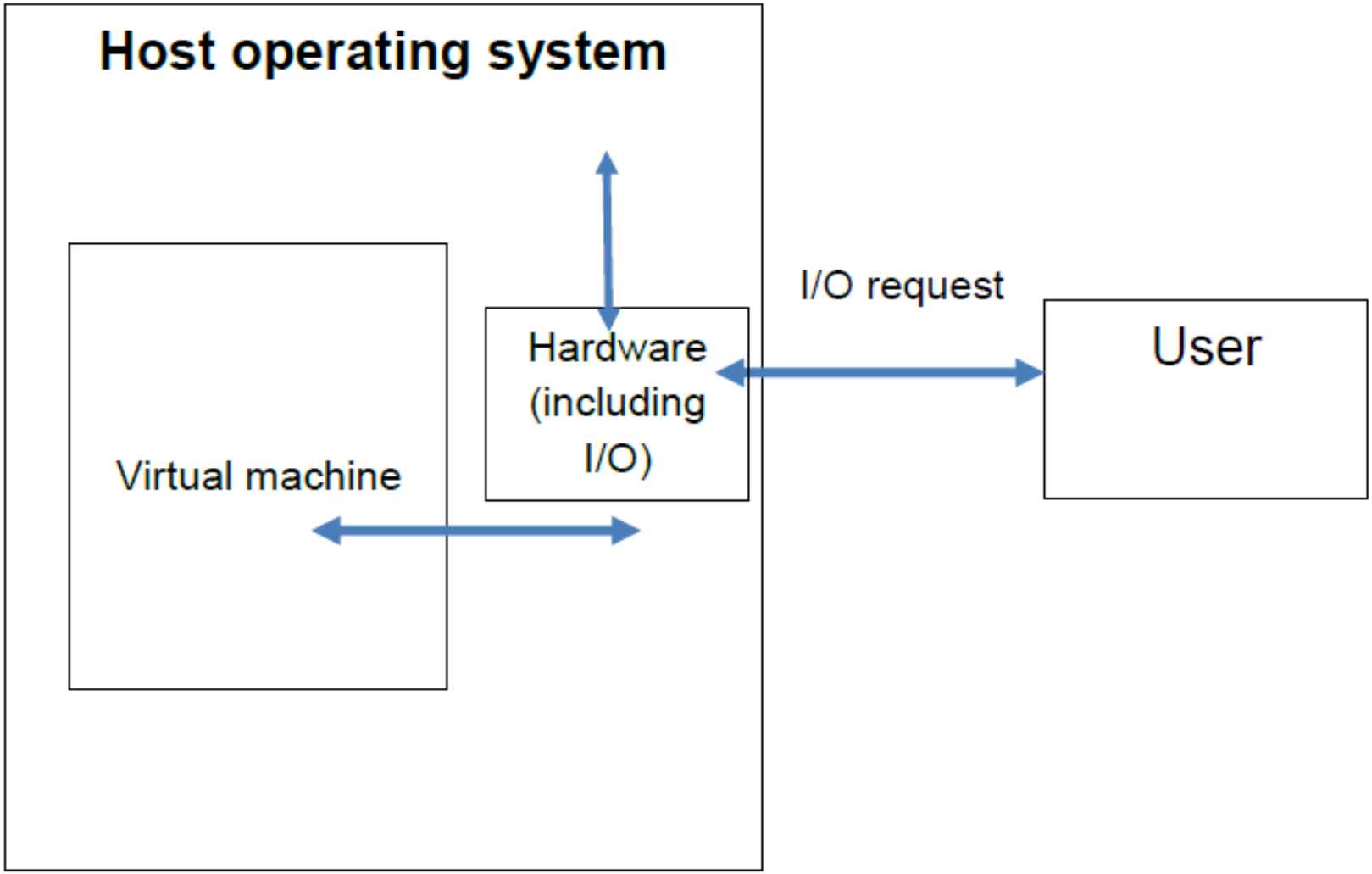
- One example would be a programmer is developing an application that needs to be launched to the entire workforce for use.The application will pull data from other sources and the users will be able to run reports for specific information.

- The programmer has to ensure the application will work on the organizations computers without any bugs.

- Before the programmer launches the application to the workforce, they have a VM loaded onto the server that has a copy of the OS that the organization uses and their standard for all their computers.

- The programmer will launch their application in the VM OS to see how it works and will fix any bugs they run into before fully launching a bug free application to the workforce. They would do the same if they have to perform any updates to the system or the application, they would test the application in the VM environment before releasing it to the workforce.

Virtual Machine

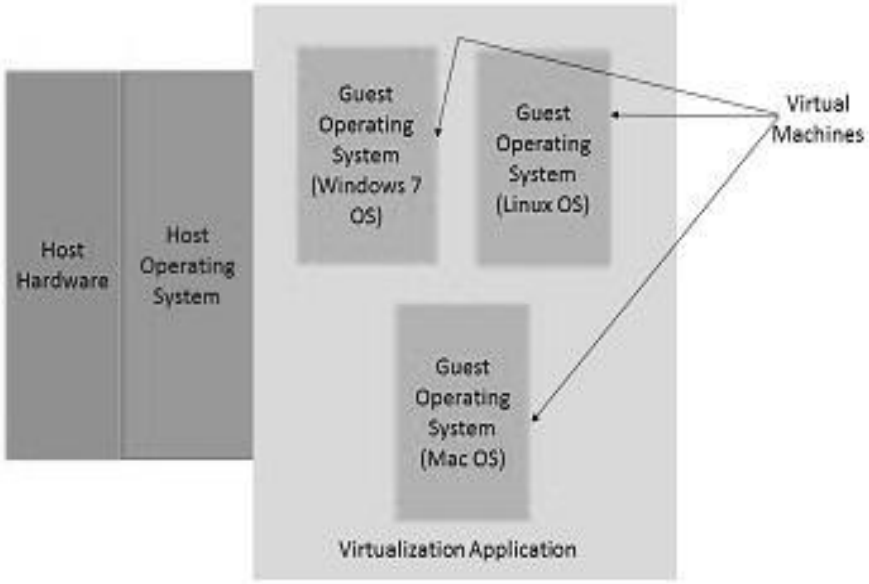


System Virtual Machines

- A system virtual machine is an environment that allows multiple instances of the operating system(VMs) to run on a host system, sharing the physical resources.

Process Virtual Machines

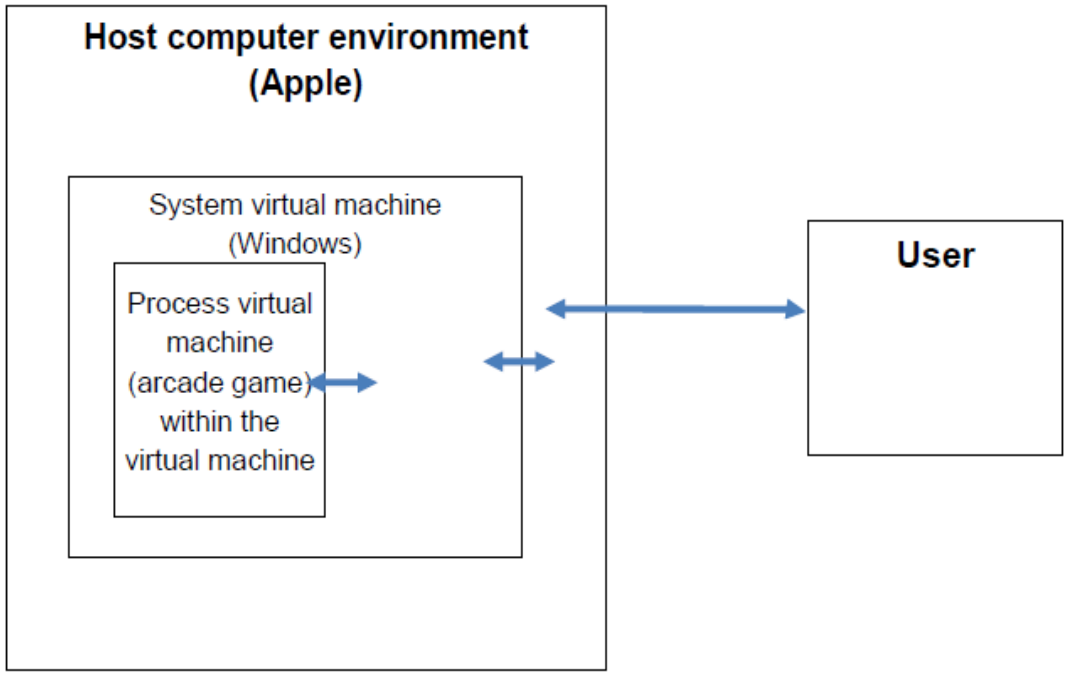
- A process virtual machine, also known as an application VM, is used to execute computer programs in a platform-independent environment. It is designed to run applications in the same way irrespective of the platform.



Virtual Machines within Virtual Machines

- It is possible to run one virtual machine within another. For example, a user wanting to play an emulated old arcade game could run the arcade game process virtual machine, within a Windows system virtual machine, on an Apple-based computer.

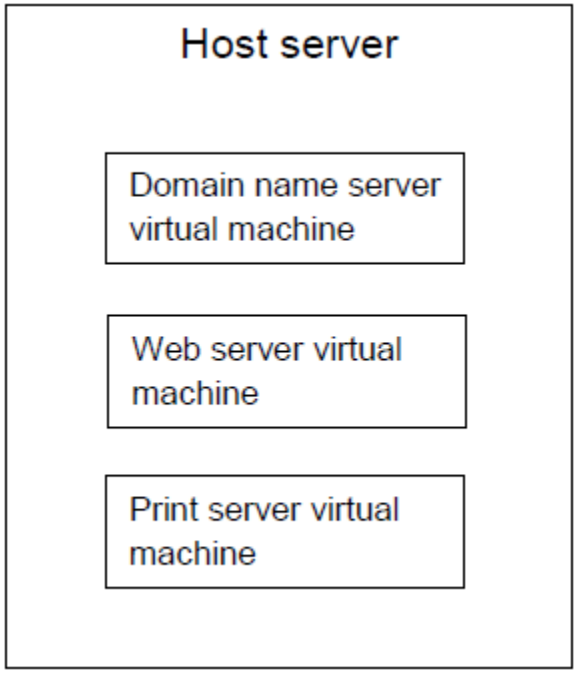
- The host computer's operating system provides an interface to the guest virtual machine, which in turn provides an interface to the process virtual machine within it.



Note: The number of virtual machines within virtual machines is limited only by the hardware's performance and available memory of the host computer.

Running Virtual Machines Simultaneously

- If a host system is powerful enough, several virtual machines can be run side-by-side on one computer. A good example of this is the increasing use of virtual machines to run several virtual servers, such as a domain name server, web server and print server, on one physical server. The host server maintains the virtual machines and allows them to act as though they are all separate computers.



Note: The number of virtual machines within virtual machines is limited only by the hardware's performance and available memory of the host computer.

Virtual Machine Manager (VMM):

- VMM also called a “hypervisor,” this is one of many hardware virtualization techniques that allow multiple operating systems, termed guests, to run concurrently on a host computer.

- It is so named because it is conceptually one level higher than a supervisory program. The hypervisor presents to the guest operating systems a virtual operating platform and manages the execution of the guest operating systems.

- Multiple instances of a variety of operating systems may share the virtualized hardware resources.

- Hypervisors are installed on server hardware whose only task is to run guest operating systems. Non-hypervisor virtualization systems are used for similar tasks on dedicated server hardware, but also commonly on desktop, portable, and even handheld computers.

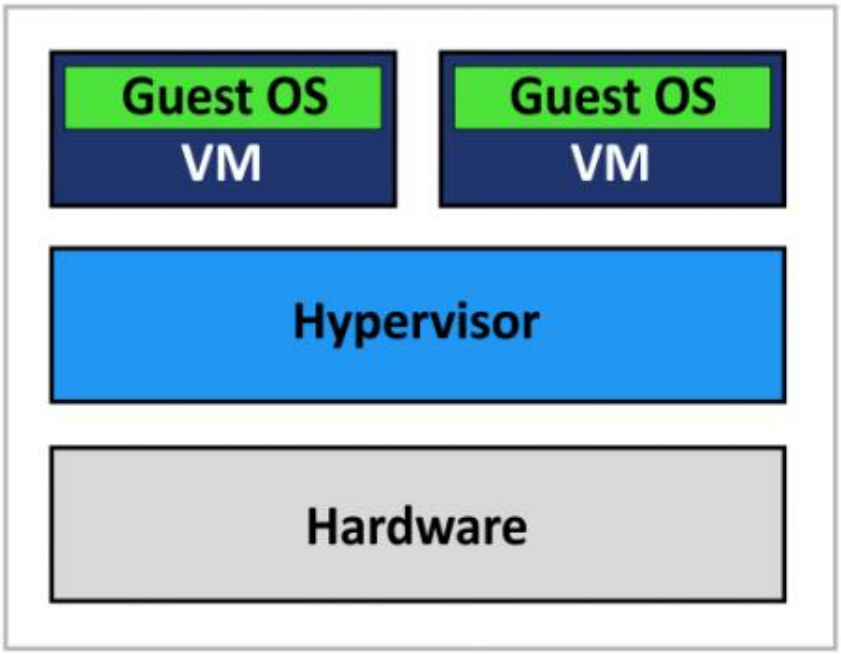
- The term is often used to describe the interface provided by the specific cloud-computing functionality infrastructure as a service (IaaS).

There are two types of hypervisors.

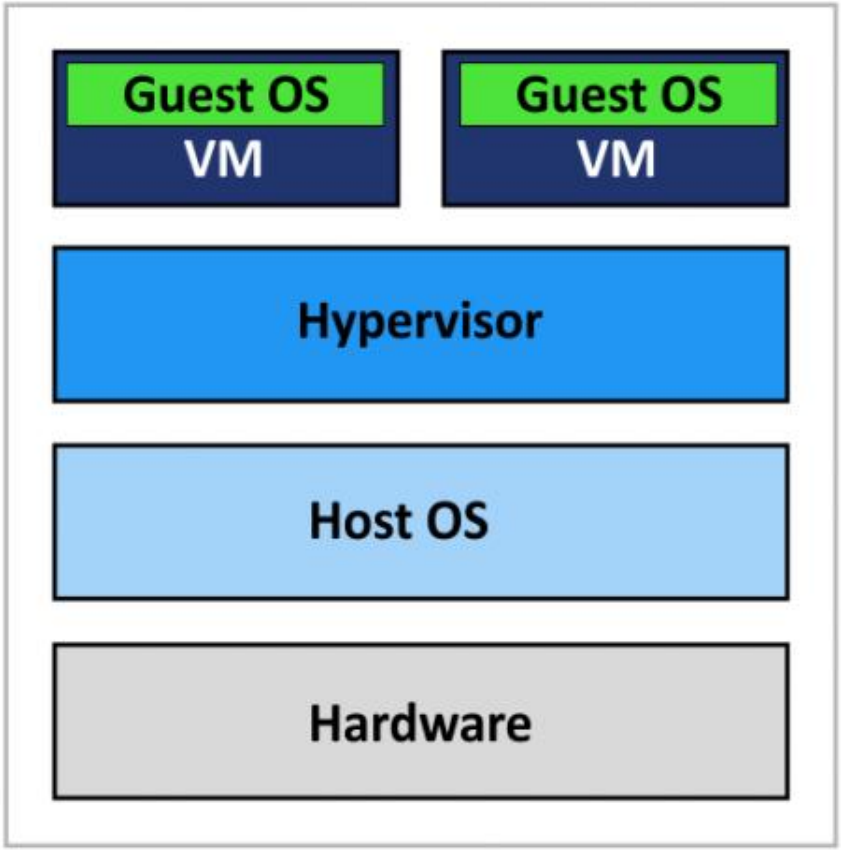
- Type 1 hypervisor is run directly on the host computer's hardware.
- Type 2 hypervisor runs from the operating system of the host machine and provides a virtual operating system that still supports input/output (I/O) and memory management functions.

Working of Hypervisor

- The servers would need to execute the hypervisor. The hypervisor, in turn, loads the client operating systems of the virtual machines.
- The hypervisor allocates the correct CPU resources, memory, bandwidth and disk storage space for each virtual machine.
- A virtual machine can create requests to the hypervisor through a variety of methods, including API calls.



Type 1 Hypervisor
(Bare-Metal Architecture)



Type 2 Hypervisor
(Hosted Architecture)

Different Types Virtual Machine Programs

Virtual Box: (Windows, Linux, Mac OS X): VirtualBox is very popular because it's open-source and completely free. There's no paid version of VirtualBox, so you don't have to deal with the usual "upgrade to get more features" upsells and nags. VirtualBox works very well, particularly on Windows and Linux where there's less competition, making it a good place to start with VMs.

VMware Player : (Windows, Linux): VMware has their own line of virtual machine programs. You can use VMware Player on Windows or Linux as a free, basic virtual machine tool. More advanced features—many of which are found in VirtualBox for free—require upgrading to the paid VMware Workstation program. We recommend starting out with VirtualBox, but if it doesn't work properly you may want to try VMware Player.

VMware Fusion : (Mac OS X): Mac users must buy VMware Fusion to use a VMware product, since the free VMware Player isn't available on a Mac. However, VMware Fusion is more polished.

Parallel Desktop : (Mac OS X): Macs also have Parallels Desktop available. Both Parallels Desktop and VMware Fusion for Mac are more polished than the virtual machine programs on other platforms, since they're marketed to average Mac users who might want to run Windows software.

Benefits and Limitations of Virtual Machines

- Using virtual machines brings several benefits:
  - They allow modern systems to use programs that run on obsolete or unsupported platforms
  - They allow platform-independent programs to run on multiple platforms
  - They allow one computer to behave as several computers simultaneously
  - They can be easily backed up, copied and re-installed. This has particular benefits in corporate environments where network downtime can lead to loss of revenue.
  - The virtual machines can be installed on another computer and be up and running again quickly.

- Virtual machines also have limitations:
  - They emulate software, not hardware. Users trying to get the full functionality from software for another platform may find that they still cannot use the software correctly. For example, an emulation of a driving video game that requires a steering wheel might not be able to operate the game correctly
  - Emulation requires processing power and memory. Running a virtual machine may take a substantial amount of these resources, resulting in reduced performance from the host computer
  - If the host system lacks sufficient resources, the virtual machine will also suffer from poor performance and be difficult to use.

LINUX

Linux Operating System

- Linux is an operating system. An operating system is software that manages all of the hardware resources associated with your desktop or laptop.

- Linux has evolved into one of the most reliable computer ecosystems on the planet. Combine that reliability with zero cost of entry and you have the perfect solution for a desktop platform.

Linux is also distributed under an open source license. Open source follows the following key philosophies:

  - The freedom to run the program, for any purpose.

  - The freedom to study how the program works, and change it to make it do what you wish.

  - The freedom to redistribute copies so you can help your neighbor.

  - The freedom to distribute copies of your modified versions to others.

Linux Operating System -Pieces

- **The Boot loader:** The software that manages the boot process of your computer. For most users, this will simply be a splash screen that pops up and eventually goes away to boot into the operating system.

- **The kernel:** This is the one piece of the whole that is actually called “Linux”. The kernel is the core of the system and manages the CPU, memory, and peripheral devices. The kernel is the “lowest” level of the OS.

- **Daemons:** These are background services (printing, sound, scheduling, etc) that either start up during boot, or after you log into the desktop.

- **The Shell:** You’ve probably heard mention of the Linux command line. This is the shell – a command process that allows you to control the computer via commands typed into a text interface. This is what, at one time, scared people away from Linux the most (assuming they had to learn a seemingly archaic command line structure to make Linux work). This is no longer the case. With modern desktop Linux, there is no need to ever touch the command line.

Linux Operating System –Pieces Cont..,

- **Graphical Server:** This is the sub-system that displays the graphics on your monitor. It is commonly referred to as the X server or just “X”.

- **Desktop Environment:** This is the piece of the puzzle that the users actually interact with. There are many desktop environments to choose from (Unity, GNOME, Cinnamon, Enlightenment, KDE, XFCE, etc). Each desktop environment includes built-in applications (such as file managers, configuration tools, web browsers, games, etc).

- **Applications:** Desktop environments do not offer the full array of apps. Just like Windows and Mac, Linux offers thousands upon thousands of high-quality software titles that can be easily found and installed. Most modern Linux distributions (more on this in a moment) include App Store-like tools that centralize and simplify application installation. For example: Ubuntu Linux has the Ubuntu Software Center (Figure 1) which allows you to quickly search among the thousands of apps and install them from one centralized location.

Linux Distributions

- Standard, precompiled sets of packages, or *distributions,* include the basic Linux system, system installation and management utilities, and ready-to-install packages of common UNIX tools.
- The first distributions managed these packages by simply providing a means of unpacking all the files into the appropriate places; modern distributions include advanced package management.
- Early distributions included SLS and Slackware. *Red Hat* and *Debian are popular distributions from commercial* and noncommercial sources, respectively.
- The RPM Package file format permits compatibility among the various Linux distributions.

Linux Distributions

The most popular Linux distributions are:

- Ubuntu Linux
- Linux Mint
- Arch Linux
- Deepin
- Fedora
- Debian
- OpenSUSE

Enterprise Distributions:

- Red Hat Enterprise Linux
- Ubuntu Server
- CentOS
- SUSE Enterprise Linux

Design Principles

Linux is a multiuser, multitasking system with a full set of UNIX-compatible tools.. Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model.

- Main design goals are speed, efficiency, and standardization.
- Linux is designed to be compliant with the relevant POSIX documents; at least two Linux distributions have achieved official POSIX certification.
- The Linux programming interface adheres to the SVR4
- UNIX semantics, rather than to BSD behavior.

Basic Features

Following are some of the important features of Linux Operating System.

- **Portable** − Portability means software can works on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.

- **Open Source** − Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

- **Multi-User** − Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.

- **Multiprogramming** − Linux is a multiprogramming system means multiple applications can run at same time.

- **Hierarchical File System** − Linux provides a standard file structure in which system files/ user files are arranged.

- **Shell** − Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.

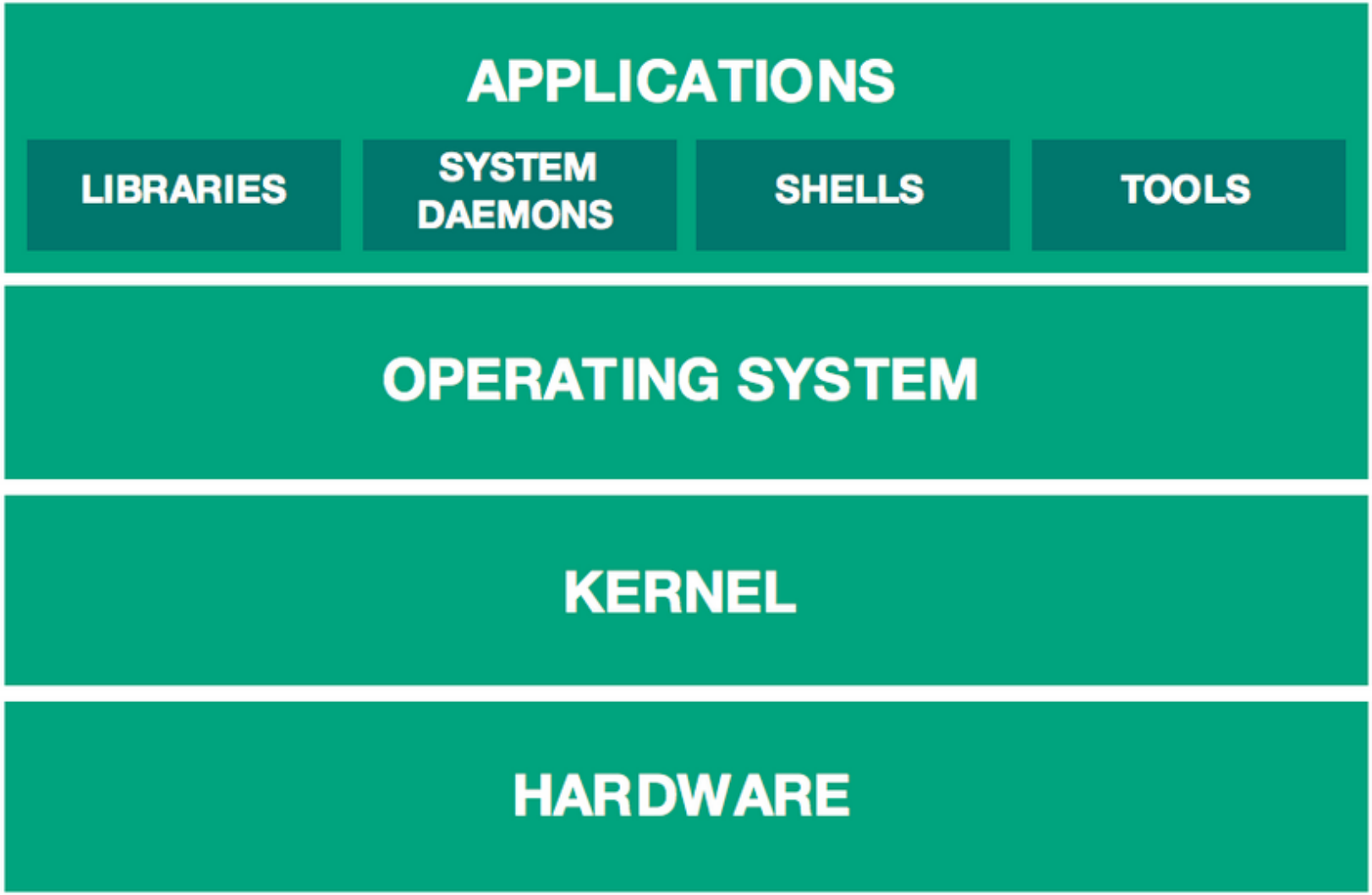
- **Security** − Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Linux Architecture

The architecture of a Linux System consists of the following layers −

- **Hardware layer** − Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** − It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** − An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
- **Utilities** − Utility programs that provide the user most of the functionalities of an operating systems.

Linux Architecture



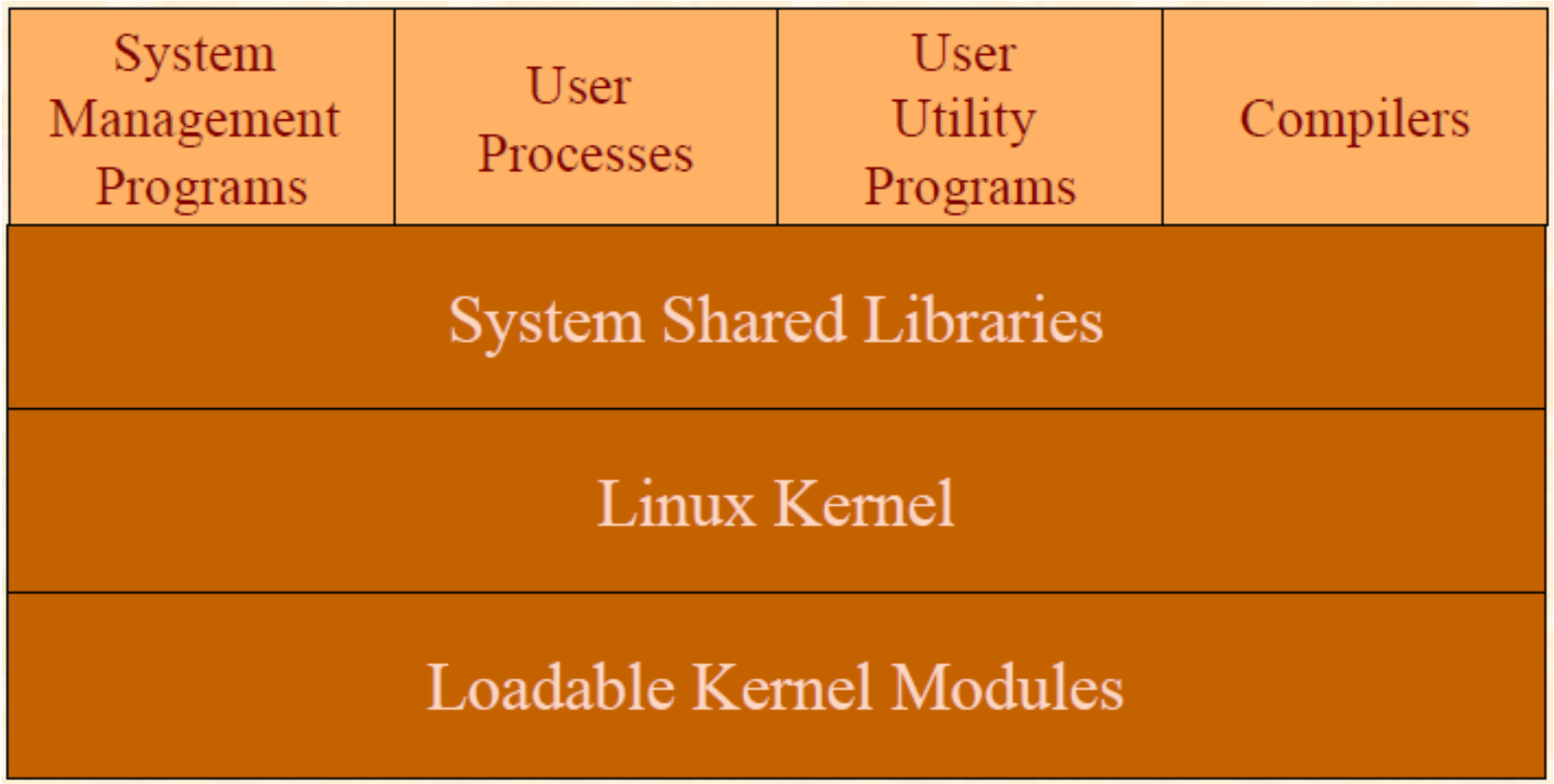
Components of a Linux System

Like Unix it has three main constituents :

- Kernel
- System libraries
- Utilities.
  - Kernel is the core that manages processes and virtual memory.
  - System libraries define functions that applications use to seek
  - kernel services without exercising the kernel code privileges.

The utilities are specialized functions like “sort” or programs, called daemons like login daemons or network connection management daemons.

Components of a Linux System



The Kernel

The kernel is the central component of a computer operating systems. The only job performed by the kernel is to the manage the communication between the software and the hardware. A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one.

Types of Kernels

There are many types of kernels that exists, but among them, the two most popular kernels are:

1**.Monolithic**

A monolithic kernel is a single code or block of the program. It provides all the required services offered by the operating system. It is a simplistic design which creates a distinct communication layer between the hardware and software.

2. Microkernels

Microkernel manages all system resources. In this type of kernel, services are implemented in different address space. The user services are stored in user address space, and kernel services are stored under kernel address space. So, it helps to reduce the size of both the kernel and operating system.

The Kernel : Operational Features

- The kernel code executes in privileged mode called kernel mode.
- Any code that does not need to run in privileged mode is put in the system library.
- The interesting thing about Linux kernel is that it has a modular architecture – even with binary codes: Linux kernel can load (and unload) dynamically (at run time) modules just it can load or unload the system library modules.

Linux Kernel : Features

- Kernel code provides for arbitrations and for protected access to HW resources.
- Kernel supports services for the applications through the system libraries. System calls within applications may also use system library. For instance, the buffered file handling is operated and managed by Linux kernel through system libraries.
- Programs like utilities that are needed to initialize the system and configure network devices are classed as user mode programs and do not run with kernel privileges (unlike in Unix).
- Programs like those that handle login requests are run as system utilities and also do not require kernel privileges (unlike in Unix).

The Kernel Module

- The “loadable” kernel modules execute in the privileged kernel mode – and therefore have the capabilities to communicate with all of HW.
- Linux kernel source code is free. People may develop their own kernel modules. However, this requires recompiling, linking and loading. Such a code can be distributed under GPL.
- More often the modality is: Start with the standard minimal basic kernel module. Then enrich the environment by the addition of customized drivers. This is the route presently most people in the embedded system area are adopting worldwide.

Module Support in Linux Kernel

- The module support in Linux has three main components :
  - Module management.
  - Driver registration.
  - Conflict resolution mechanism.

Module Management

Supports loading modules into memory and letting them talk to the rest of the kernel. Module loading is split into two separate sections:

- Managing sections of module code in kernel memory

- Handling symbols that modules are allowed to reference The module requestor manages loading requested, but currently unloaded, modules; it also regularly queries the kernel to see whether a dynamically loaded module is still in use, and will unload it when it is no longer actively needed.

Driver Registration

- Allows modules to tell the rest of the kernel that a new driver has become available.

The kernel maintains dynamic tables of all known drivers, and provides a set of routines to allow drivers to be added to or removed from these tables at any time.

- Registration tables include the following items:
  - Device drivers
  - File systems
  - Network protocols
  - Binary format

Process Environment

The process's environment is inherited from its parent, and is composed of two null-terminated vectors:

- The argument vector lists the command-line arguments used to invoke the running program; conventionally starts with the name of the program itself
- The environment vector is a list of "NAME=VALUE" pairs that associates named environment variables with arbitrary textual values.
  - Passing environment variables among processes and inheriting variables by a process's children are flexible means of passing information to components of the user mode system software.
  - The environment-variable mechanism provides a customization of the operating system that can be set on a per-process basis, rather than being configured for the system as a whole.

Process Context

The (constantly changing) state of a running program at any point in time.

- The **scheduling context is the most important part of the** process context; it is the information that the scheduler needs to suspend and restart the process.

- The kernel maintains **accounting information about the** resources currently being consumed by each process, and the total resources consumed by the process in its lifetime so far.

- The **file table is an array of pointers to kernel file** structures. When making file I/O system calls, processes refer to files by their index into this table. Whereas the file table lists the existing open files, the

- **file-system context applies to requests to open new** files. The current root and default directories to be used for new file searches are stored here.

- The **signal-handler table defines the routine in the** process's address space to be called when specific signals arrive.

- The **virtual-memory context of a process describes the** full contents of the its private address space.

Processes and Threads

- Linux uses the same internal representation for processes and threads; a thread is simply a new process that happens to share the same address space as its parent.
- A distinction is only made when a new thread is created by the **clone system call.**
  - **Fork creates a new process with its own entirely new** process context
  - **Clone creates a new process with its own identity, but that is** allowed to share the data structures of its parent Using **clone gives an application fine-grained control over** exactly what is shared between two threads.

Scheduling

- The job of allocating CPU time to different tasks within an operating system.
- While scheduling is normally thought of as the running and interrupting of processes, in Linux, scheduling also includes the running of the various kernel tasks.
- Running kernel tasks encompasses both tasks that are requested by a running process and tasks that execute internally on behalf of a device driver.

Kernel Synchronization

A request for kernel-mode execution can occur in two ways:

- A running program may request an operating system service, either explicitly via a system call, or implicitly, for example, when a page fault occurs.
- A device driver may deliver a hardware interrupt that causes the CPU to start executing a kernel-defined handler for that interrupt.
- Kernel synchronization requires a framework that will allow the kernel's critical sections to run without interruption by another critical section.

Kernel Synchronization cont.

Linux uses two techniques to protect critical sections:

- Normal kernel code is non preemptible
  - when a time interrupt is received while a process is executing a kernel system service routine, the kernel's **need_resched flag is set so that the scheduler will run** once the system call has completed and control is about to be returned to user mode.
- The second technique applies to critical sections that occur in an interrupt service routines.
  - By using the processor's interrupt control hardware to disable interrupts during a critical section, the kernel guarantees that it can proceed without the risk of concurrent access of shared data structures.

To avoid performance penalties, Linux's kernel uses a synchronization architecture that allows long critical sections to run without having interrupts disabled for the critical section's entire duration.

Interrupt service routines are separated into a *top half* and a *bottom half.*

- The top half is a normal interrupt service routine, and runs with recursive interrupts disabled.
- The bottom half is run, with all interrupts enabled, by a miniature scheduler that ensures that bottom halves never interrupt themselves.
- This architecture is completed by a mechanism for disabling selected bottom halves while executing normal, foreground kernel code.

Directories Found On Linux Systems

Directory	Comments
/	The root directory. Where everything begins.
/bin	Contains binaries (programs) that must be present for the system to boot and run.
/boot	Contains the Linux kernel, initial RAM disk image (for drivers needed at boot time), and the boot loader. Interesting files: ▪ /boot/grub/grub.conf or menu.lst, which are used to configure the boot loader. ▪ /boot/vmlinuz, the Linux kernel
/dev	This is a special directory which contains device nodes. “Everything is a file” also applies to devices. Here is where the kernel maintains a list of all the devices it understands.

Directory	Comments
/etc	The /etc directory contains all of the system-wide configuration files. It also contains a collection of shell scripts which start each of the system services at boot time. Everything in this directory should be readable text. Interesting files: While everything in /etc is interesting, here are some of my all-time favorites: ▪ /etc/crontab, a file that defines when automated jobs will run. ▪ /etc/fstab, a table of storage devices and their associated mount points. ▪ /etc/passwd, a list of the user accounts.
/home	In normal configurations, each user is given a directory in /home. Ordinary users can only write files in their home directories. This limitation protects the system from errant user activity.
/lib	Contains shared library files used by the core system programs. These are similar to DLLs in Windows.
/lost+found	Each formatted partition or device using a Linux file system, such as ext3, will have this directory. It is used in the case of a partial recovery from a file system corruption event. Unless something really bad has happened to your system, this directory will remain empty.

Directory	Comments
/media	On modern Linux systems the /media directory will contain the mount points for removable media such as USB drives, CD-ROMs, etc. that are mounted automatically at insertion.
/mnt	On older Linux systems, the /mnt directory contains mount points for removable devices that have been mounted manually.
/opt	The /opt directory is used to install “optional” software. This is mainly used to hold commercial software products that may be installed on your system.
/proc	The /proc directory is special. It's not a real file system in the sense of files stored on your hard drive. Rather, it is a virtual file system maintained by the Linux kernel. The “files” it contains are peepholes into the kernel itself. The files are readable and will give you a picture of how the kernel sees your computer.
/root	This is the home directory for the root account.
/sbin	This directory contains “system” binaries. These are programs that perform vital system tasks that are generally reserved for the superuser.

Directory	Comments
/tmp	The /tmp directory is intended for storage of temporary, transient files created by various programs. Some configurations cause this directory to be emptied each time the system is rebooted.
/usr	The /usr directory tree is likely the largest one on a Linux system. It contains all the programs and support files used by regular users.
/usr/bin	/usr/bin contains the executable programs installed by your Linux distribution. It is not uncommon for this directory to hold thousands of programs.
/usr/lib	The shared libraries for the programs in /usr/bin.
/usr/local	The /usr/local tree is where programs that are not included with your distribution but are intended for system wide use are installed. Programs compiled from source code are normally installed in /usr/local/bin. On a newly installed Linux system, this tree exists, but it will be empty until the system administrator puts something in it.
/usr/sbin	Contains more system administration programs.

Directory	Comments
/usr/share/doc	Most packages installed on the system will include some kind of documentation. In /usr/share/doc, we will find documentation files organized by package.
/var	With the exception of /tmp and /home, the directories we have looked at so far remain relatively static, that is, their contents don't change. The /var directory tree is where data that is likely to change is stored. Various databases, spool files, user mail, etc. are located here.
/var/log	/var/log contains log files, records of various system activity. These are very important and should be monitored from time to time. The most useful one is /var/log/messages. Note that for security reasons on some systems, you must be the superuser to view log files .

LINUX - Basic Commands

mkdir – Create Directories

The mkdir command is used to create directories. It works like this:

- **mkdir *directory...***

A note on notation: When three periods follow an argument in the description of a command (as above), it means that the argument can be repeated, thus:

- **mkdir dir1**

would create a single directory named “dir1”, while

- **mkdir dir1 dir2 dir3**

would create three directories named “dir1”, “dir2”, and “dir3”.

cp – Copy Files And Directories

The cp command copies files or directories. It can be used two different ways:

- **cp *file1 file2***

to copy the single file or directory “file1” to file or directory “file2” and:

- **cp *file... Directory***

to copy multiple files or directories into a directory.

cp Command Options

Option	Meaning
-a, --archive	Copy the files and directories and all of their attributes, including ownerships and permissions. Normally, copies take on the default attributes of the user performing the copy.
-i, --interactive	Before overwriting an existing file, prompt the user for confirmation. **If this option is not specified, cp will silently overwrite files.**
-r, --recursive	Recursively copy directories and their contents. This option (or the -a option) is required when copying directories.
-u, --update	When copying files from one directory to another, only copy files that either don't exist, or are newer than the existing corresponding files, in the destination directory.
-v, --verbose	Display informative messages as the copy is performed.

cp Command Examples

Command	Results
cp *file1 file2*	Copy *file1* to *file2*. **If *file2* exists, it is overwritten with the contents of *file1*.** If *file2* does not exist, it is created.
cp -i *file1 file2*	Same as above, except that if *file2* exists, the user is prompted before it is overwritten.
cp *file1 file2 dir1*	Copy *file1* and *file2* into directory *dir1*. *dir1* must already exist.
cp dir1/* dir2	Using a wildcard, all the files in *dir1* are copied into *dir2*. *dir2* must already exist.
cp -r *dir1 dir2*	Copy the contents of directory *dir1* to directory *dir2*. If directory *dir2* does not exist, it is created and, after the copy, will contain the same contents as directory *dir1*. If directory *dir2* does exist, then directory *dir1* (and its contents) will be copied into *dir2*.

mv Command – Move And Rename Files

- The mv command performs both file moving and file renaming, depending on how it is used. In either case, the original filename no longer exists after the operation.

mv is used in much the same way as cp:

- **mv *file1 file2***

to move or rename file or directory “item1” to “item2” or:

- **mv *file... Directory***

to move one or more items from one directory to another.

mv Command Options

Option	Meaning
-i, --interactive	Before overwriting an existing file, prompt the user for confirmation. **If this option is not specified, mv will silently overwrite files.**
-u, --update	When moving files from one directory to another, only move files that either don't exist, or are newer than the existing corresponding files in the destination directory.
-v, --verbose	Display informative messages as the move is

mv Command Examples

Command	Result
mv *file1 file2*	Move *file1* to *file2*. **If *file2* exists, it is overwritten with the contents of *file1*.** If *file2* does not exist, it is created. **In either case, *file1* ceases to exist.**
mv -i *file1 file2*	Same as above, except that if *file2* exists, the user is prompted before it is overwritten.
mv *file1 file2 dir1*	Move *file1* and *file2* into directory *dir1*. *dir1* must already exist.
mv *dir1 dir2*	If directory *dir2* does not exist, create directory *dir2* and move the contents of directory *dir1* into *dir2* and delete directory *dir1*. If directory *dir2* does exist, move directory *dir1* (and its contents) into directory *dir2*.

rm command – Remove Files And Directories

The rm command is used to remove (delete) files and directories:

- **rm *item*...**

where “item” is one or more files or directories.

rm Command Options

-i, --interactive	Before deleting an existing file, prompt the user for confirmation. **If this option is not specified, rm will silently delete files.**
-r, --recursive	Recursively delete directories. This means that if a directory being deleted has subdirectories, delete them too. To delete a directory, this option must be specified.
-f, --force	Ignore nonexistent files and do not prompt. This overrides the --interactive option.
-v, --verbose	Display informative messages as the deletion is performed.

rm Command Examples

rm *file1*	Delete *file1* silently.
rm -i *file1*	Same as above, except that the user is prompted for confirmation before the deletion is performed.
rm -r *file1 dir1*	Delete *file1* and *dir1* and its contents.
rm -rf *file1 dir1*	Same as above, except that if either *file1* or *dir1* do not exist, rm will continue silently.

Useful Tip. Whenever you use wildcards with rm (besides carefully checking your typing!), test the wildcard first with ls. This will let you see the files that will be deleted. Then press the up arrow key to recall the command and replace the ls with rm.

ln – Create Links

The ln command is used to create either hard or symbolic links. It is used in one of two ways:

ln *file link*

to create a hard link, and:

ln -s *item link*

to create a symbolic link where “item” is either a file or a directory.

type – Display A Command's Type

The type command is a shell builtin that displays the kind of command the shell will execute, given a particular command name. It works like this:

type *command*

where “command” is the name of the command you want to examine.

Queries ?

Thank You