Assignment-1

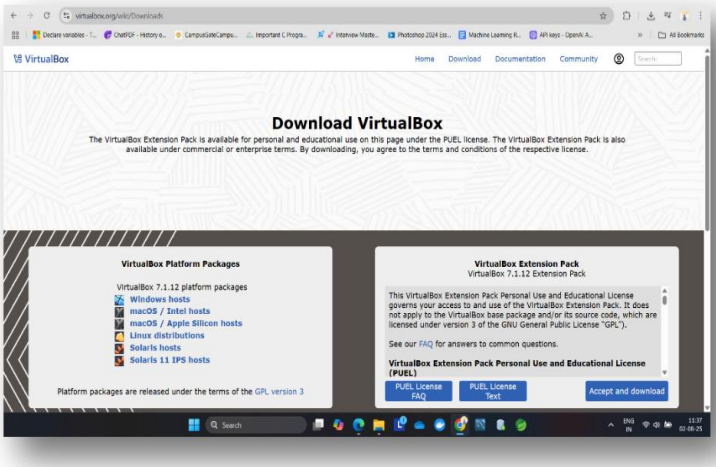
Cloud Computing

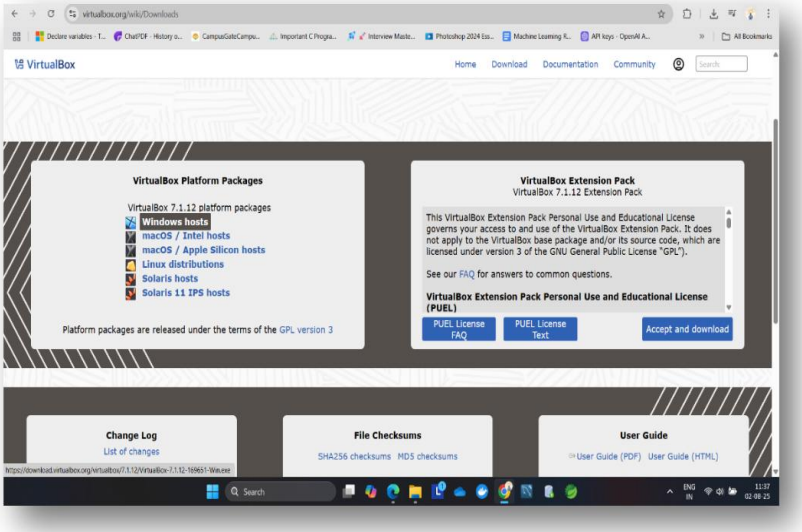
Virtualization: Installing VirtualBox & Setting Up Multiple OSes

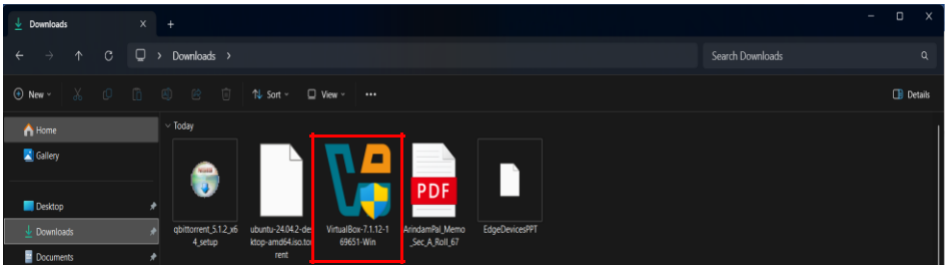
Objective: Install Oracle VirtualBox and create virtual machines (VMs) with different Linux distributions.

Step 1: Download VirtualBox

Go to: https://www.virtualbox.org
Click **Downloads**
Choose **Windows hosts** to download the `.exe` installer





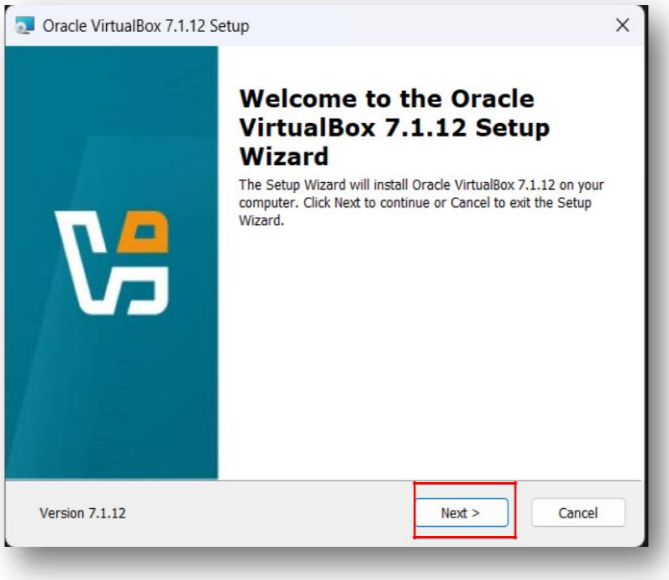


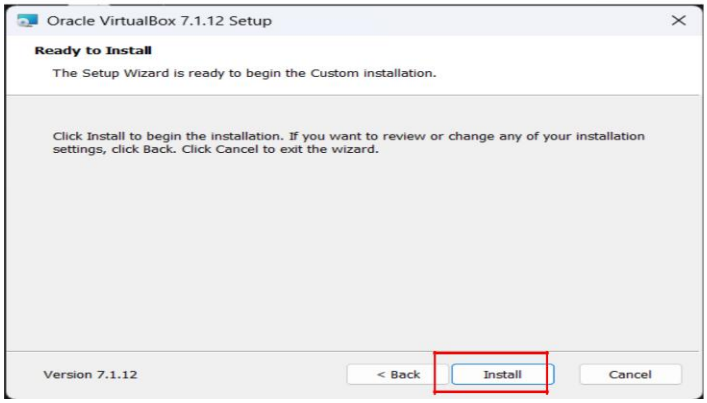
Step 2: Run the Installer

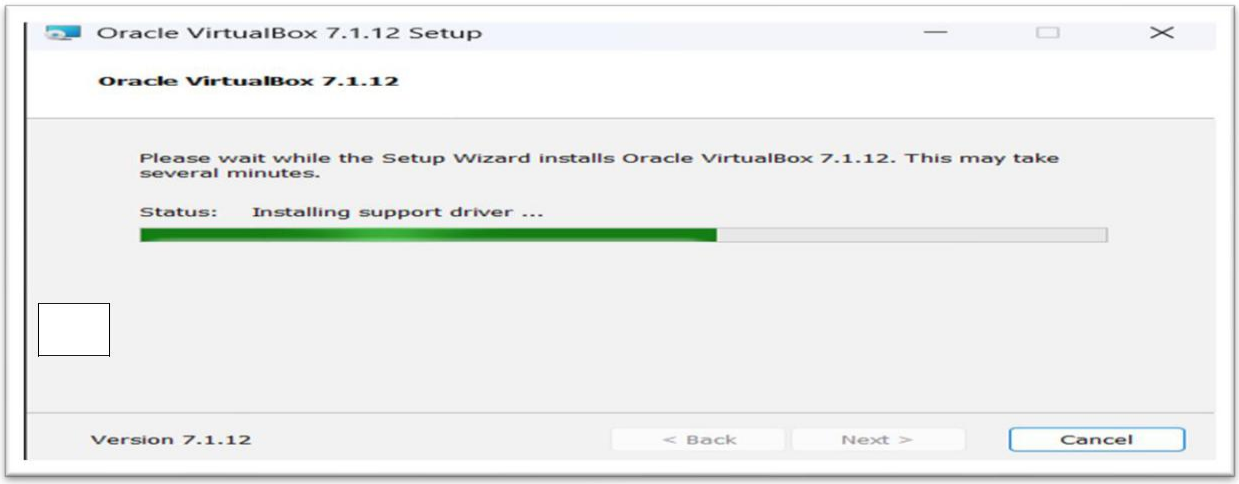
Double-click the downloaded `.exe` file

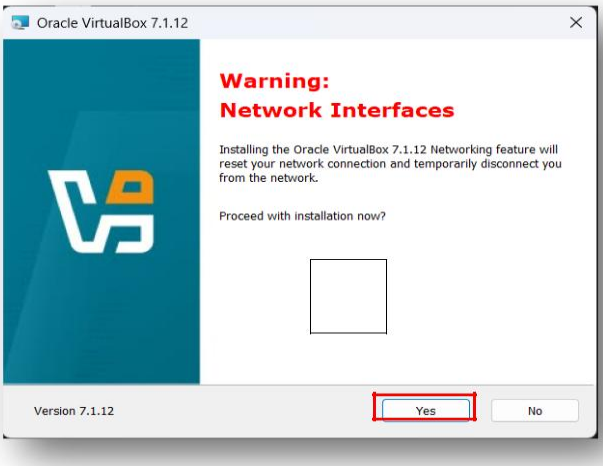
Click **Next** through the prompts (default options are fine)

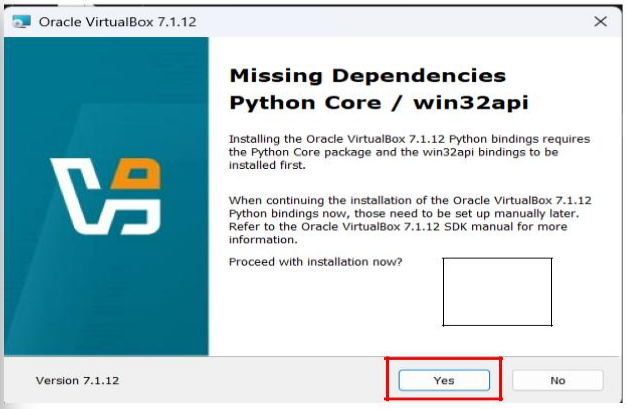
Allow network access if prompted (important for VM networking)
Click **Install**, then **Finish** when done

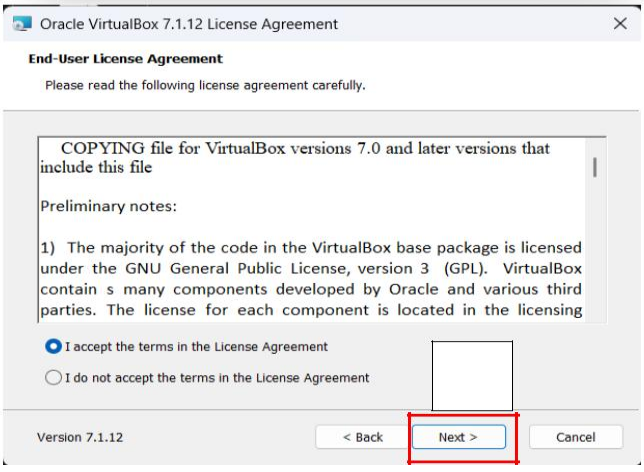


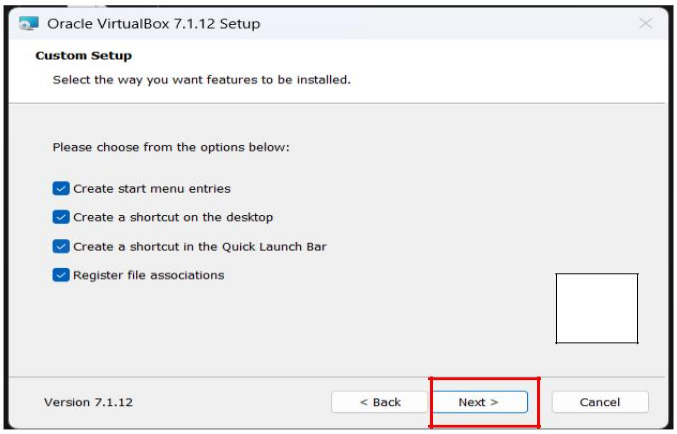


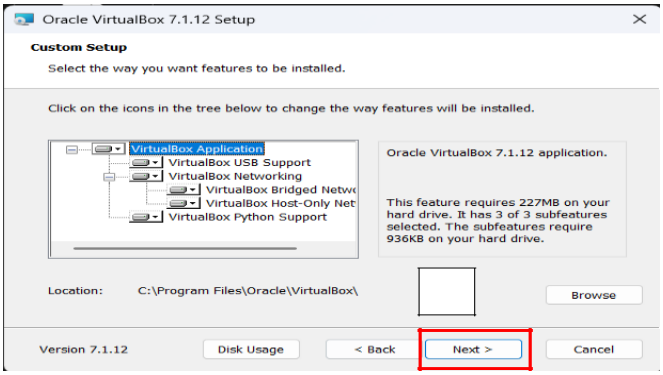

Oracle VirtualBox 7.1.12 Setup
Oracle VirtualBox 7.1.12
Please wait while the Setup Wizard installs Oracle VirtualBox 7.1.12. This may take several minutes.
Status: Installing support driver ...
Version 7.1.12
< Back
Next >
Cancel

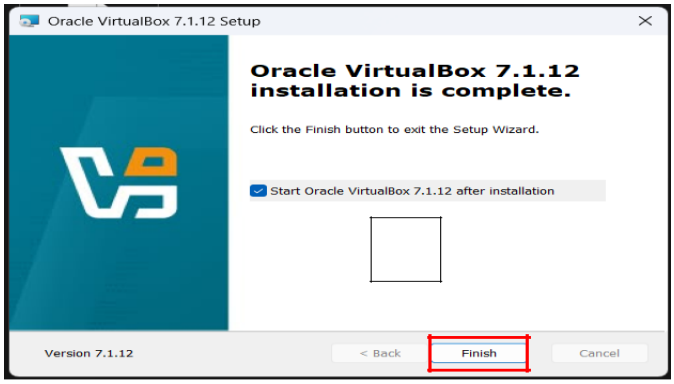

Oracle VirtualBox 7.1.12
Warning:
Network Interfaces
Installing the Oracle VirtualBox 7.1.12 Networking feature will reset your network connection and temporarily disconnect you from the network.
Proceed with installation now?
Version 7.1.12
Yes
No


Oracle VirtualBox 7.1.12
Missing Dependencies
Python Core / win32api
Installing the Oracle VirtualBox 7.1.12 Python bindings requires the Python Core package and the win32api bindings to be installed first.
When continuing the installation of the Oracle VirtualBox 7.1.12 Python bindings now, those need to be set up manually later. Refer to the Oracle VirtualBox 7.1.12 SDK manual for more information.
Proceed with installation now?
Version 7.1.12
Yes
No


Oracle VirtualBox 7.1.12 License Agreement
End-User License Agreement
Please read the following license agreement carefully.
COPYING file for VirtualBox versions 7.0 and later versions that include this file
Preliminary notes:
1) The majority of the code in the VirtualBox base package is licensed under the GNU General Public License, version 3 (GPL). VirtualBox contain s many components developed by Oracle and various third parties. The license for each component is located in the licensing
I accept the terms in the License Agreement
I do not accept the terms in the License Agreement
Version 7.1.12
< Back
Next >
Cancel


Oracle VirtualBox 7.1.12 Setup
Custom Setup
Select the way you want features to be installed.
Please choose from the options below:
Create start menu entries
Create a shortcut on the desktop
Create a shortcut in the Quick Launch Bar
Register file associations
Version 7.1.12
< Back
Next >
Cancel

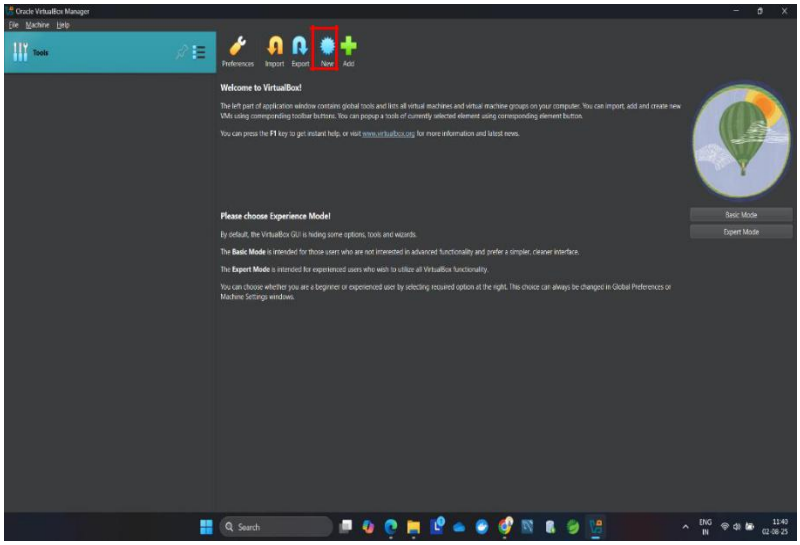

Oracle VirtualBox 7.1.12 Setup
Custom Setup
Select the way you want features to be installed.
Click on the icons in the tree below to change the way features will be installed.
VirtualBox Application
VirtualBox USB Support
VirtualBox Networking
VirtualBox Bridged Netw
VirtualBox Host-Only Net
VirtualBox Python Support
Oracle VirtualBox 7.1.12 application.
This feature requires 227MB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 936KB on your hard drive.
Location: C:\Program Files\Oracle\VirtualBox\
Browse
Version 7.1.12
Disk Usage
< Back
Next >
Cancel

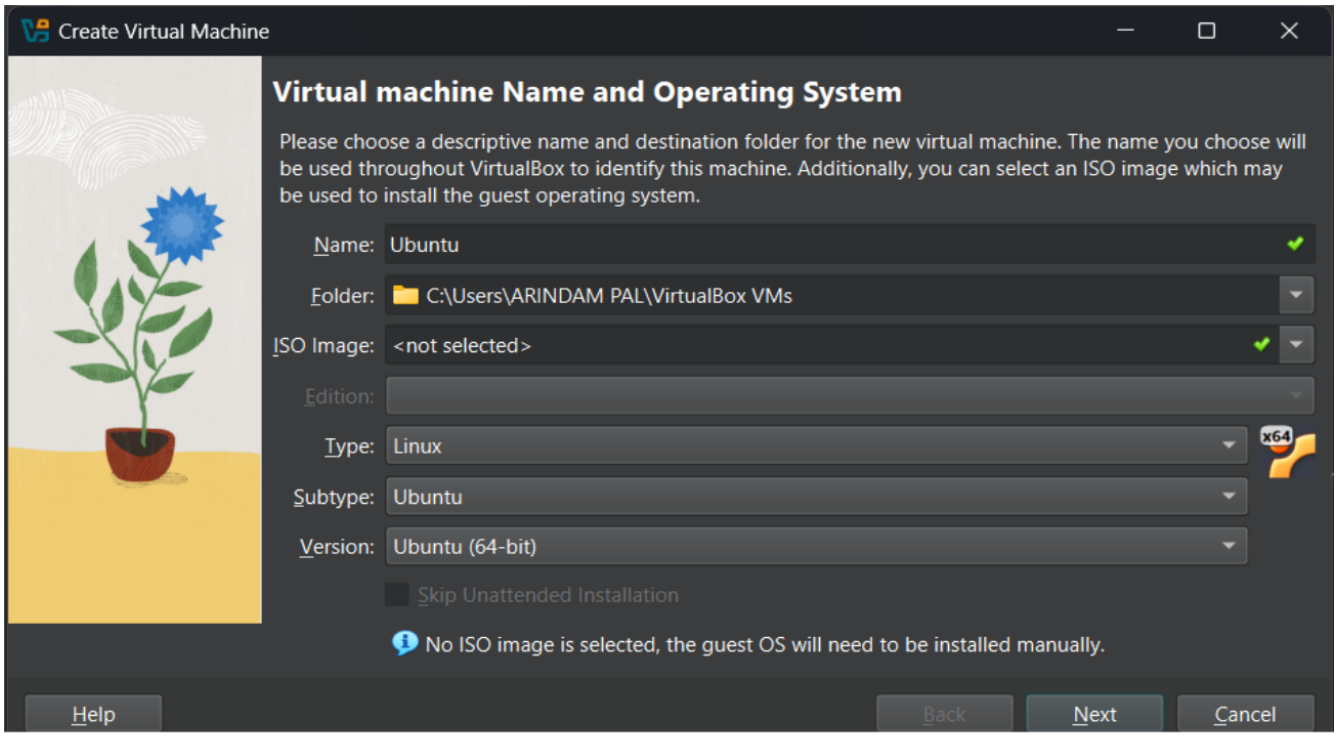

Oracle VirtualBox 7.1.12 Setup
Oracle VirtualBox 7.1.12 installation is complete.
Click the Finish button to exit the Setup Wizard.
Start Oracle VirtualBox 7.1.12 after installation
Version 7.1.12
< Back
Finish
Cancel

Create a New Virtual Machine

Step 1: Open VirtualBox and Create New VM

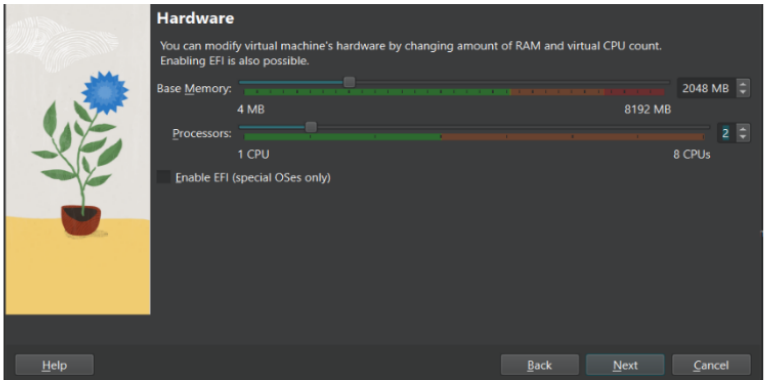
- Click **"New"**
- Name: `Ubuntu`
- Type: **Linux**
- Version: **Ubuntu (64-bit)**
- Click **Next**





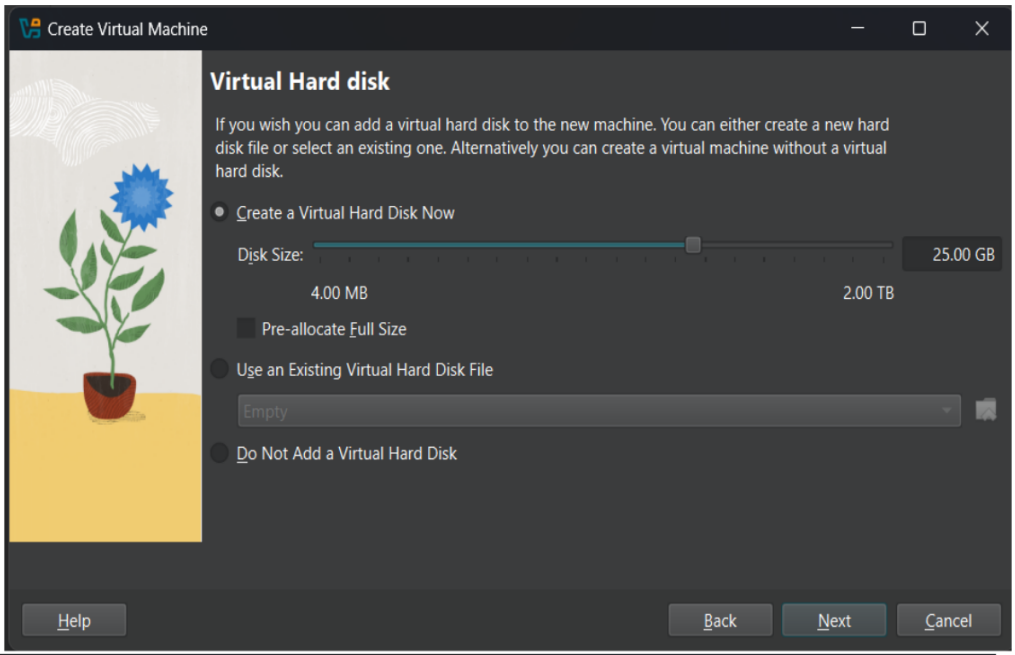
Step 2: Set Memory Size (RAM)

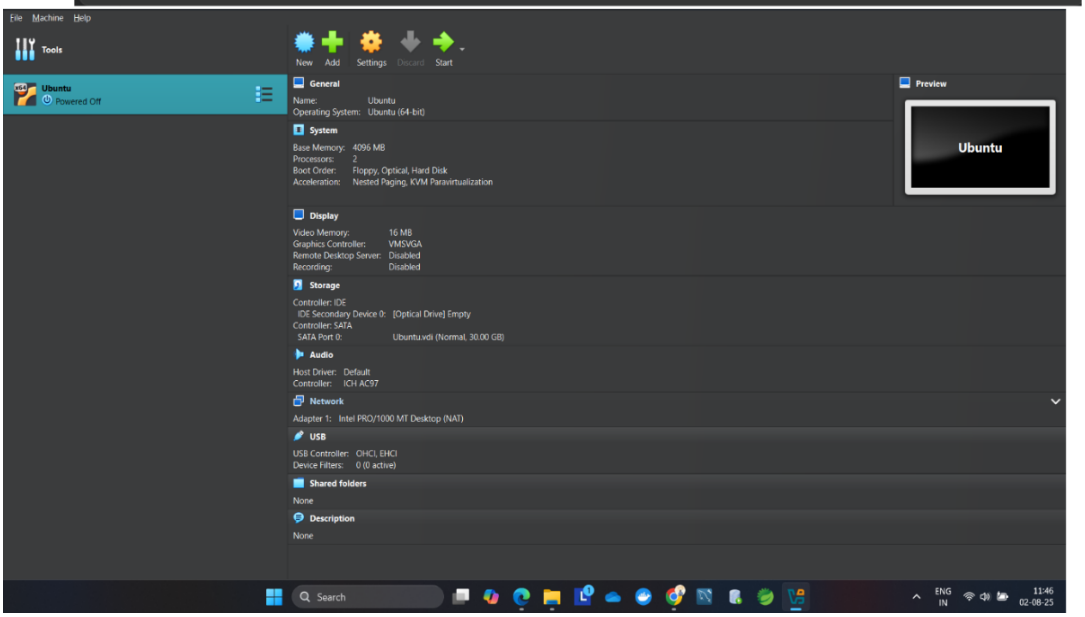
- Choose at least **2048 MB (2 GB)** for lightweight use
- **4096 MB (4 GB)** is better if your system has 8+ GB RAM



Step 3: Create Virtual Hard Disk

- Select: **Create a virtual hard disk now**
- Click **Create**
- Choose **VDI** (VirtualBox Disk Image) → Click **Next**
- Choose **Dynamically allocated** → Click **Next**
- Set disk size to **25–40 GB** → Click **Create**



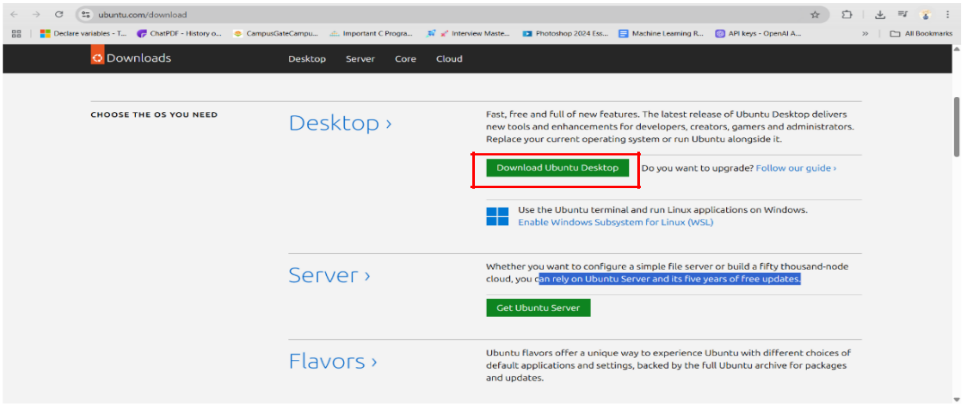


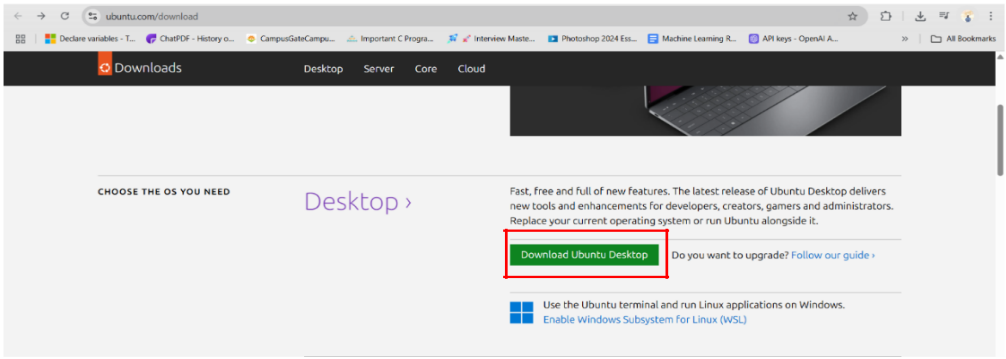
Download Ubuntu ISO

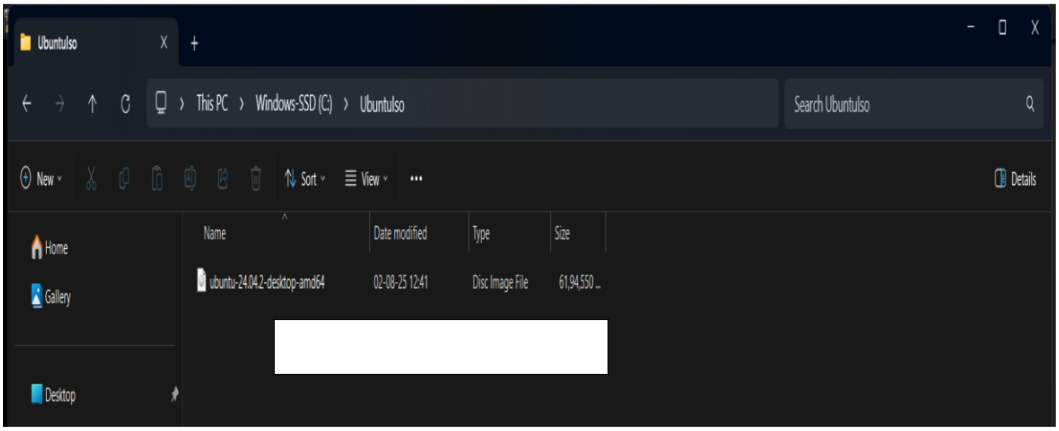
Step 1: Get Ubuntu Desktop ISO

- Go to: https://ubuntu.com/download/desktop
- Click **Download Ubuntu Desktop**

Wait for the **~5.9 GB** ISO file to download







Downloaded File

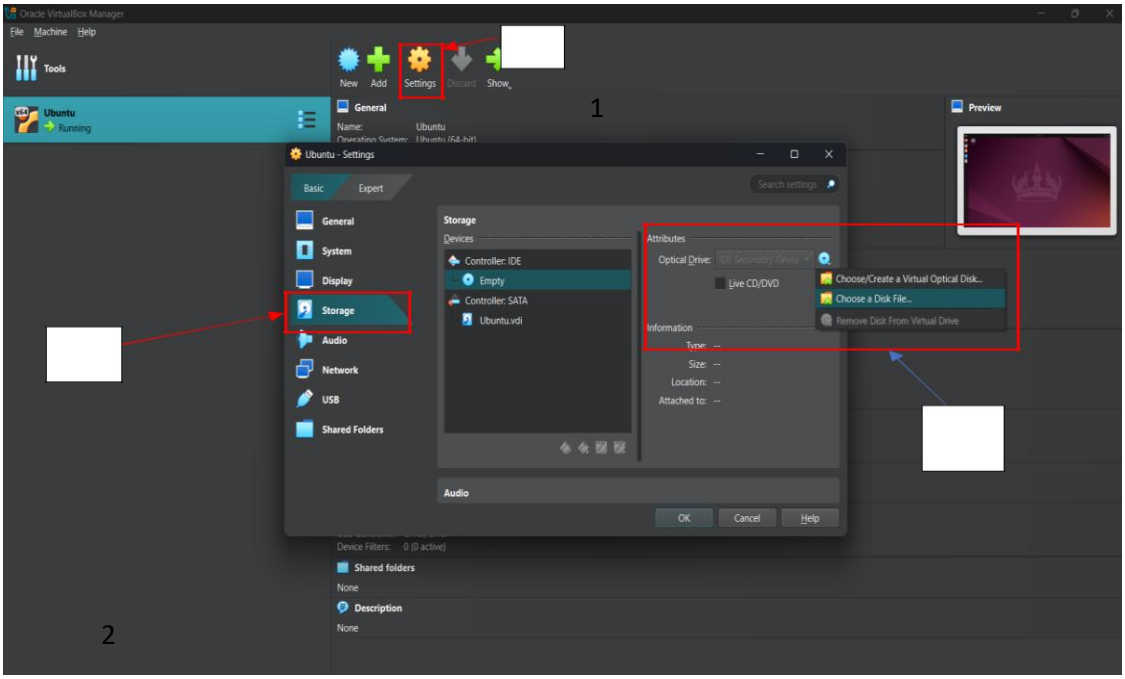
Attach Ubuntu ISO

Step 1: Go to VM Settings → Storage

- Under **Controller: IDE**, click the empty CD icon
- On the right, click the CD icon again → Choose **"Choose a disk file…"**
- Browse and select the Ubuntu ISO you downloaded

Step 2: Save and Start

- Click **OK**
- Click **Start** on your VM



Install Ubuntu OS in the VM

Step 1: Select "Try or Install Ubuntu"

- When prompted in the black screen (GRUB menu), press **Enter** on: **Try or Install Ubuntu**

Step 2: Choose "Install Ubuntu"

- Select language → Click **Install Ubuntu**
- Keyboard layout → Click **Continue**

Step 3: Installation Options

- Choose **Normal Installation**
- Select **Erase disk and install Ubuntu**
  *(Don't worry, this erases the **virtual** hard disk only, not your real system)*
- Click **Install Now** → Confirm if prompted

Step 4: Set User Info

- Set your name, computer name, username, and password
- Click **Continue**

Step 5: Wait for Installation

- Let it complete (may take 5–10 minutes)

Step 6: Reboot

- When it says **Installation complete**, click **Restart Now**
- **Important**: Before reboot, remove the ISO
- Go to **Devices → Optical Drives → Remove Disk from Virtual Drive** ○ Then press **Enter** to reboot

Done

ASSIGNMENT 3

Objective

Simulate cloud scenarios and implement a custom scheduling algorithm in CloudSim.

Tools Required

1. **Eclipse IDE**
2. **CloudSim 3.0.3**
3. **Java Development Kit (JDK)** – Version 8 or later

Procedure & Implementation

Step 1: Setup CloudSim

1. Install **JDK** and configure JAVA_HOME.
2. Install **Eclipse IDE**.
3. Download and extract **CloudSim-3.0.3**.
4. Open Eclipse → Create a new **Java Project** → Add CloudSim `.jar` files from `cloudsim-3.0.3/jars/` into **Build Path**.

Step 2: Simulation Code

```
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;

import java.util.*;

public class BasicExample {
    public static void main(String[] args) {
        int numUsers = 1;   // number of cloud users
        Calendar calendar = Calendar.getInstance();
        boolean traceFlag = false;
```

```
        CloudSim.init(numUsers, calendar, traceFlag);

        Datacenter datacenter0 = createDatacenter("Datacenter_0");

        DatacenterBroker broker = createBroker();
        int brokerId = broker.getId();

        Cloudlet cloudlet = createCloudlet(brokerId);
        Vm vm = createVM(brokerId);

        broker.submitVmList(List.of(vm));
        broker.submitCloudletList(List.of(cloudlet));

        CloudSim.startSimulation();
        CloudSim.stopSimulation();

        List<Cloudlet> results = broker.getCloudletReceivedList();
        for (Cloudlet cl : results) {
            System.out.println("Cloudlet " + cl.getCloudletId() + " finished with status " +
cl.getStatus());
        }
    }

    // Helper methods for Datacenter, Broker, VM, and Cloudlet creation
    private static Datacenter createDatacenter(String name) {
        List<Host> hostList = new ArrayList<>();
        List<Pe> peList = new ArrayList<>();
        peList.add(new Pe(0, new PeProvisionerSimple(1000))); // one CPU

        hostList.add(new Host(0, new RamProvisionerSimple(2048),
                new BwProvisionerSimple(10000), 1000000, peList,
                new VmSchedulerTimeShared(peList)));

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                "x86", "Linux", "Xen", hostList, 10.0, 3.0,
                0.05, 0.001, 0.0);

        try {
            return new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), new LinkedList<>(), 0);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    private static DatacenterBroker createBroker() {
        try {
```

```
            return new DatacenterBroker("Broker");
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    private static Vm createVM(int brokerId) {
        return new Vm(0, brokerId, 1000, 1, 1024, 1000, 10000,
                "Xen", new CloudletSchedulerTimeShared());
    }

    private static Cloudlet createCloudlet(int brokerId) {
        UtilizationModel utilization = new UtilizationModelFull();
        return new Cloudlet(0, 40000, 1, 300, 300, utilization, utilization, utilization);
    }
}
```

Step 3: Scheduling Algorithm

Example: **Deadline-Based Scheduling**

- Modify **VmScheduler** to prioritize tasks with earlier deadlines.

```
public class DeadlineVmScheduler extends VmSchedulerTimeShared {

    public DeadlineVmScheduler(List<Pe> peList) {

        super(peList);

    }

    @Override

    public double updateVmProcessing(double currentTime, List<Double> mipsShare)
{

        // Sort cloudlets based on deadline (custom logic)

        // Prioritize cloudlets with nearest deadline

        return super.updateVmProcessing(currentTime, mipsShare);

    }

}
```

Stimulation Outputs

========= Simulation Results =========

Cloudlet 1 finished with status SUCCESS on VM 0 | Deadline: 10.0 | Finish Time: 5.0

Cloudlet 0 finished with status SUCCESS on VM 0 | Deadline: 20.0 | Finish Time: 12.0

Cloudlet 2 finished with status SUCCESS on VM 0 | Deadline: 30.0 | Finish Time: 20.0

ASSIGNMENT 2

Objective

Install GCC (or Clang) and run simple C programs on a Linux VM.

Tools

- `build-essential` (GCC, make)
- Text editor: `nano`, `vim`, or an IDE (CodeLite/NetBeans)

Key Commands

```
sudo apt update
sudo apt install -y build-essential
gcc --version

# compile
gcc hello.c -o hello
./hello

# with warnings
gcc -Wall -Wextra -O2 hello.c -o hello
```

Sample programs

```
#include <stdio.h>
int main(void) {
    int a, b;
    printf("Enter two integers: ");
    if (scanf("%d %d", &a, &b) == 2) {
        printf("Sum = %d\n", a + b);
    } else {
        printf("Invalid input\n");
    }
    return 0;
}
```

Results

- **The gcc compiler is successfully installed and c programs are executed with it.**