BIT

BAHIR DAR UNIVERSITY

Faculty of Computing

Department: software engineering

Project title: NIXOS

Name: Elsabeth Eyayu

ID:BDU1601363

Section: A

Submitted to:Wendimu Baye

Submission Date:16/08/2017E.C

Table of Contents

1) Introduction

NixOS is an open source and free linux distribution system. NixOS history began in the early 2000s. It was initially a research project of Eelco Dolstra in the Netherlands. He wanted to solve common Linux problems like corrupt updates and messy package installations. In 2003, he created the Nix package manager, and in 2006, NixOS was introduced as a complete operating system built around it. NixOS has developed over time into a powerful and multi-purpose OS used by developers, researchers, and system administrators around the world.

For the project, I was given a Nixos, Linux system which is very different from what we are familiar with, like Ubuntu or Fedora. The project is to install Nixos with virtual machine and get familiar with it automatically, and then do some programming at the system level with it, like implementing a system call like ftruncate().

NixOS does things differently when it comes to the system. Instead of installing software and adjusting settings one by one, you specify everything your system should include in a single file. From that file, the system builds itself. This is known as declarative configuration and this makes managing the system easier, for example, if there's something broken, or someone is working on multiple machines. It also supports atomic updates, which means any modification or update occurs in one go and if something goes wrong, you can simply roll back to the previous version like an "undo" button.

While I did not choose NixOS myself, it was an assignment for the class , I think it's a great learning experience. Not only is it teaching me how to install an OS, but how to think about system configuration in a straightforward and organized way. It's also giving me real exposure to virtualization and system programming, skills that any software engineer can use.

1.1) Objectives

The main objectives of this project are:

- ✓ To establish NixOS in a virtual environment (for example, VirtualBox), along with correct installation procedures and how the process can be documented.
- ✓ To study the unique configuration model of NixOS, for example, how declarative system configurations and atomic updates work.
- ✓ To study and outline the concept of virtualization in modern operating systems and how it is achieved with tools like VirtualBox.
- ✓ In order to invoke a system call (ftruncate(), here) and learn how the operating system functions, system calls are used.
- ✓ In order to detect issues met in the installation or operation of NixOS and note possible remedies.
- ✓ In order to learn basic system programming in coding and compilation of simple programs using the C language, which operate system-level facilities.
- ✓ To have actual knowledge about the handling of software by the operating systems, filesystems, and processes.

2) Requirements

Hardware (Virtual Machine Configuration)

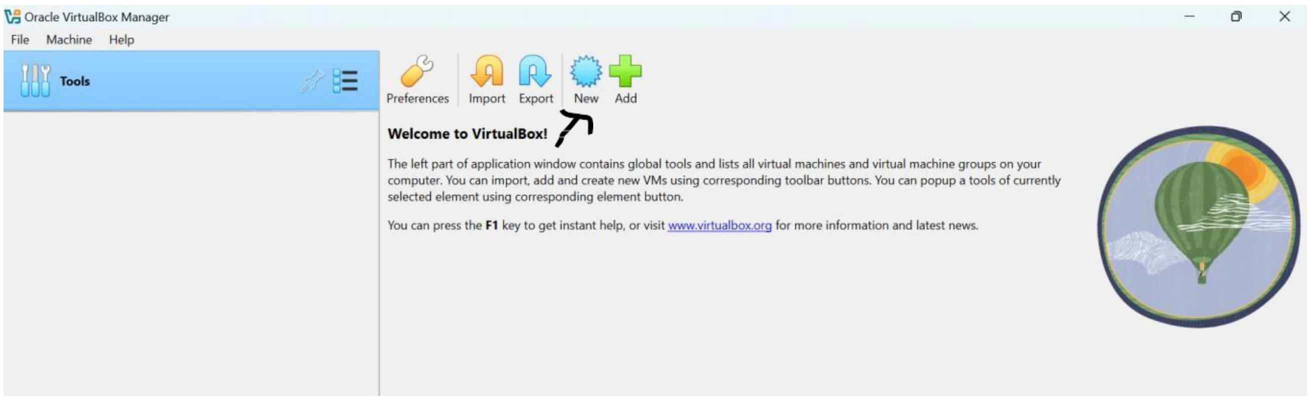
- Virtual Hard Disk: 20 GB
- Base Memory (RAM): 1024 MB (2 GB)
- Processor: 2 CPUs

Software

- Virtualization Tool: VirtualBox
- OS Image: NixOS ISO (Graphical Installer or Minimal ISO)

3) Installation steps

❖ Open virtual box and click New button in the upper part

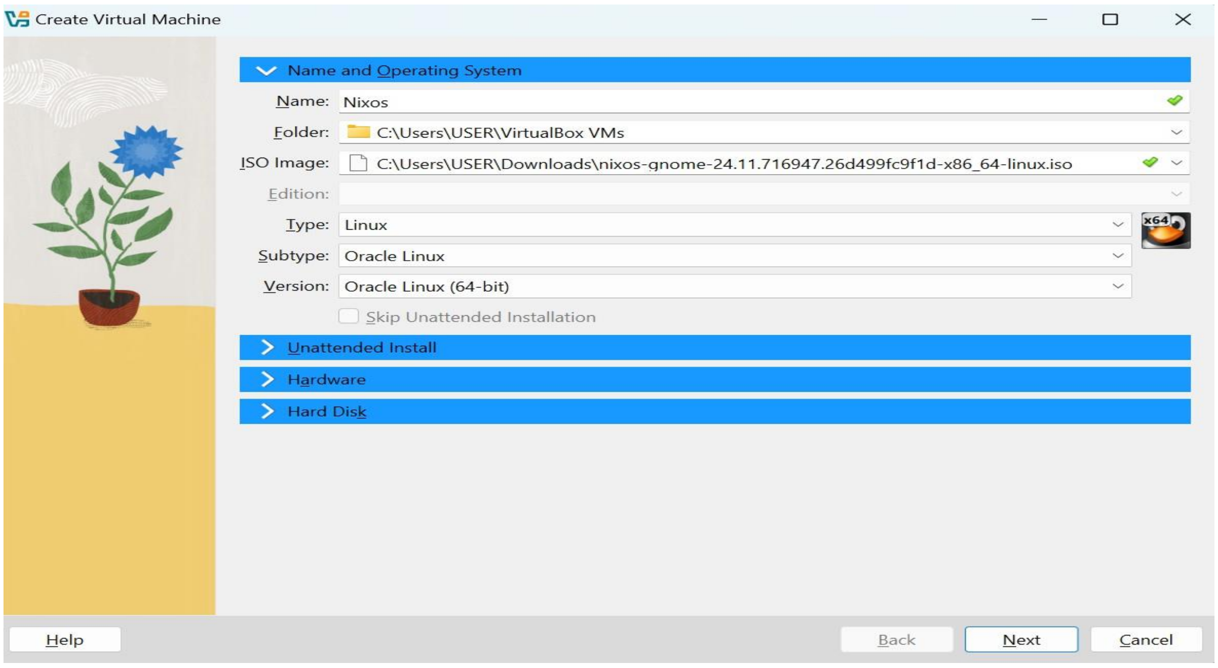


Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

❖ To initiate NixOS installation, a new virtual machine is created in VirtualBox. The VM is labeled "Nixos" in the installation dialog, and the ISO image nixos-gnome-24.11.716947.26d499fc9f1d-x86_64-linux.iso downloaded above is selected as the boot image. The type of the operating system is selected as "Linux" with subtype and version both as "Oracle Linux (64-bit)" for compatibility.


Create Virtual Machine

Name and Operating System

Name: Nixos

Folder: C:\Users\USER\VirtualBox VMs

ISO Image: C:\Users\USER\Downloads\nixos-gnome-24.11.716947.26d499fc9f1d-x86_64-linux.iso

Edition:

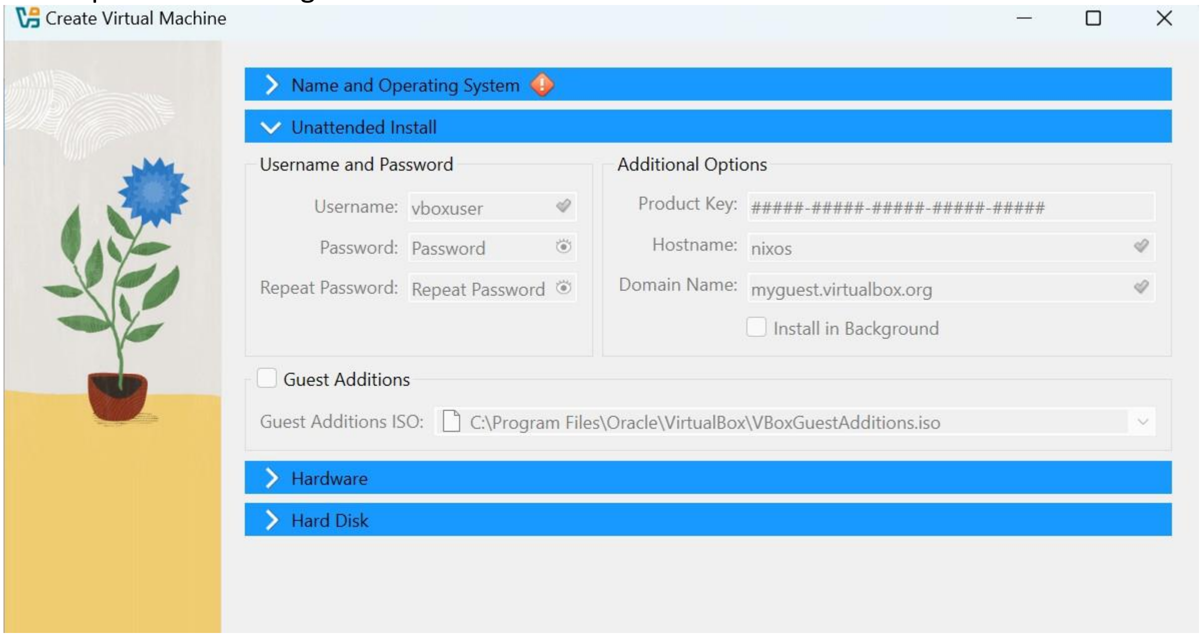
Type: Linux

Subtype: Oracle Linux

Version: Oracle Linux (64-bit)

Skip Unattended Installation

❖ The "Unattended Install" option is intentionally skipped to perform a manual installation of NixOS. In this section, the user can automatically set up username, password, hostname, domain, and Guest Additions. However, since NixOS must be installed in a way that is not a standard unattended installation, all fields in this section are left untouched. The checkbox for "Guest Additions" is also left unselected, since it is not required at this stage.


Create Virtual Machine

Name and Operating System

Unattended Install

Username and Password

Username: vboxuser

Password: Password

Repeat Password: Repeat Password

Additional Options

Product Key: #####-#####-#####-#####-#####

Hostname: nixos

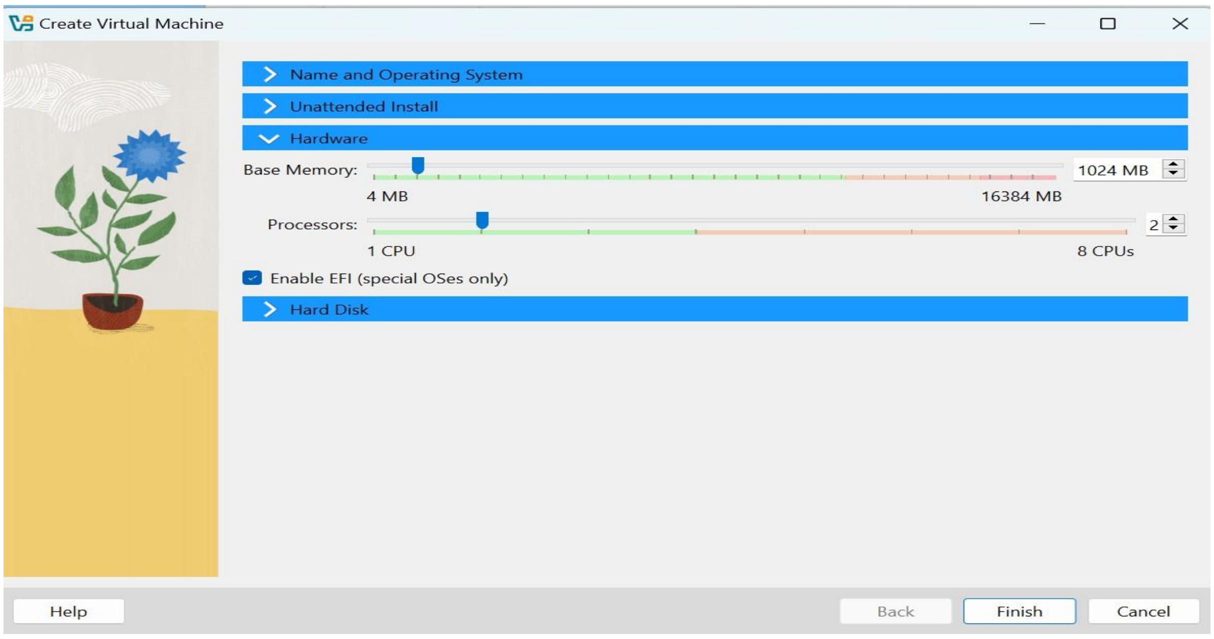
Domain Name: myguest.virtualbox.org

Install in Background

Guest Additions

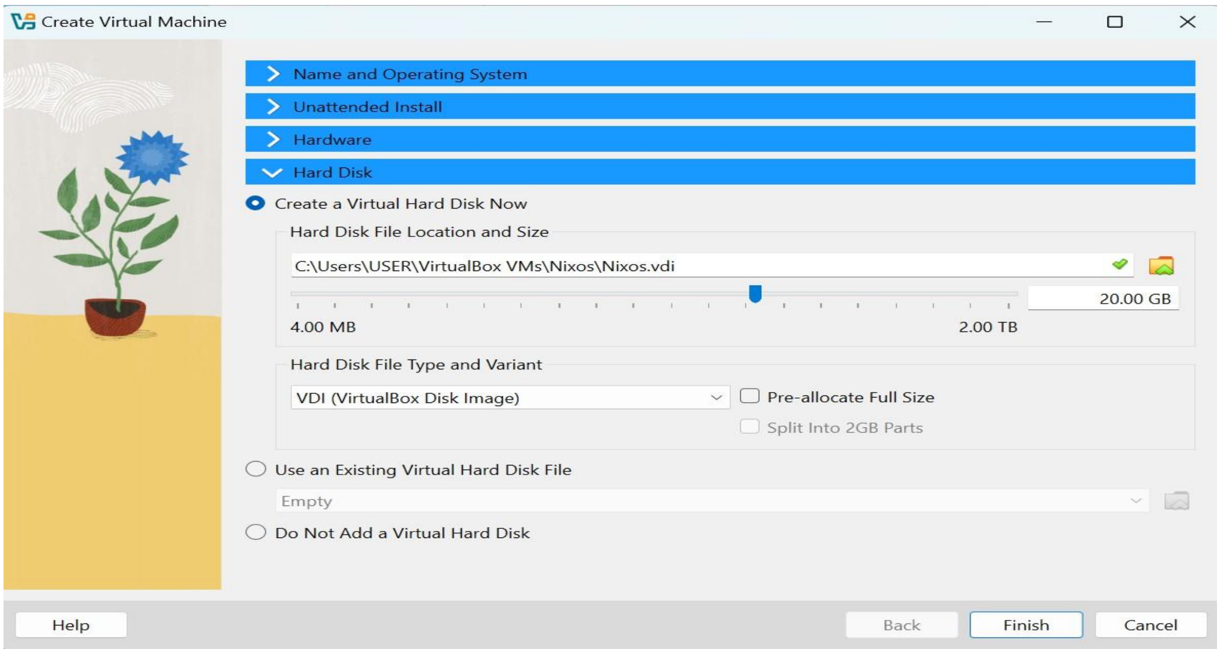
Guest Additions ISO: C:\Program Files\Oracle\VirtualBox\VBoxGuestAdditions.iso

❖ In the hardware option, set at least the base memory in 1024MB and processor at 2 CPU. I enabled EFI to use the modern boot process, as most current systems rely on UEFI. It offers faster boot times, better partitioning with GPT, and better hardware compatibility. EFI also ensures NixOS works similarly to real-world systems.



Base Memory: 1024 MB (4 MB – 16384 MB)
Processors: 2 (1 CPU – 8 CPUs)
[✓] Enable EFI (special OSes only)

❖ I created a new virtual hard disk for my NixOS virtual machine using VirtualBox. I selected the option "Create a virtual hard disk now" because I wanted to start with a fresh disk. I named it nixos.vdi and saved it in the default location that VirtualBox gave me.I chose to give the disk 20 GB of storage, which is enough for installing NixOS along with a desktop environment and some extra software. I picked the VDI (VirtualBox Disk Image) format, which is the standard format used by VirtualBox.I also made sure the disk is dynamically allocated. This means it won't take up the full 20 GB on my real hard drive immediately — it will only grow in size as the virtual machine needs more space. I did not select the pre-allocate full size option, so my actual storage space is used efficiently. Then select finish button



(●) Create a Virtual Hard Disk Now

Hard Disk File Location and Size

C:\Users\USER\VirtualBox VMs\Nixos\Nixos.vdi

20.00 GB (4.00 MB – 2.00 TB)

Hard Disk File Type and Variant

VDI (VirtualBox Disk Image)

[ ] Pre-allocate Full Size

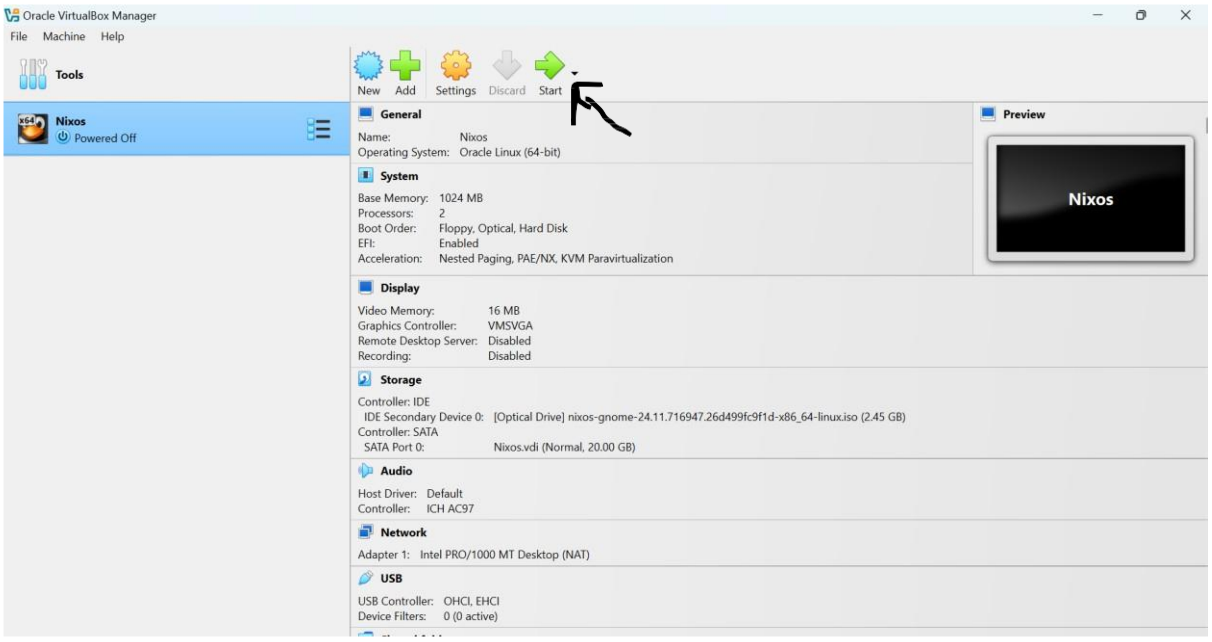
[ ] Split Into 2GB Parts

( ) Use an Existing Virtual Hard Disk File

Empty

( ) Do Not Add a Virtual Hard Disk

❖ Upon finishing, Select the start button in the page.



General
Name: Nixos
Operating System: Oracle Linux (64-bit)

System
Base Memory: 1024 MB
Processors: 2
Boot Order: Floppy, Optical, Hard Disk
EFI: Enabled
Acceleration: Nested Paging, PAE/NX, KVM Paravirtualization

Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: IDE
IDE Secondary Device 0: [Optical Drive] nixos-gnome-24.11.716947.26d499fc9f1d-x86_64-linux.iso (2.45 GB)
Controller: SATA
SATA Port 0: Nixos.vdi (Normal, 20.00 GB)

Audio
Host Driver: Default
Controller: ICH AC97

Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)

USB
USB Controller: OHCI, EHCI
Device Filters: 0 (0 active)

❖ When you boot from the NixOS installation medium (USB or ISO), you'll be presented with a menu containing several options. Each one is for a different situation, depending on your computer setup. Here's what each does, what you'll see, and when you'd use it:

1. NixOS Installer (Default Option)

- This is the standard way to boot the installer.
- You will get some loading text for a few seconds and then land in a terminal screen on a black background.
- Use this if everything is functioning properly and you just want to install NixOS.

2. Installer (nomodeset)

- This is for graphics problem-solving. It turns off some of the graphics capabilities.
- You'll land in the same terminal screen, but it's useful if the first option gave you a black screen or graphics corruption.
- Use this if the first does not display correctly.

3. Installer (copytoram)

- This copies the entire installer to your computer's memory (RAM), allowing you to eject the USB stick after booting.
- It takes longer to load but acts the same as the default when loaded.
- Use this if you need to have the USB port free after booting.

4. Installer (debug)

- This entry boots with extra technical details for debugging.
- You'll see lots more system messages and logs on the screen.
- Only use this if the installation is failing or you want to debug something.

5. HIDPI, Quirks, and Accessibility

- This opens up a small menu with options for high-resolution displays and accessibility features like a screen reader.
- Only use this if you're on a very crisp screen or you need help with visibility/accessibility.

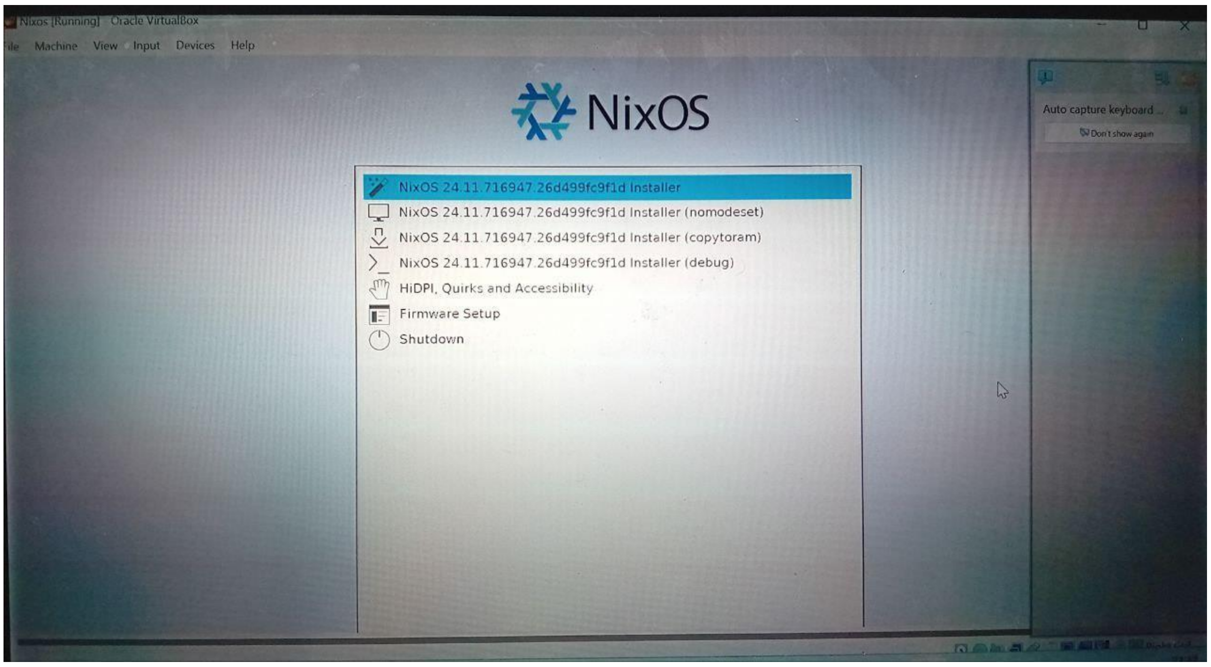
6. Firmware Setup

- This takes you to your computer's BIOS or UEFI settings.
- The screen looks more technical — boot options, secure boot setup, etc.
- Use this if you need to change your boot order or hardware settings before installing.

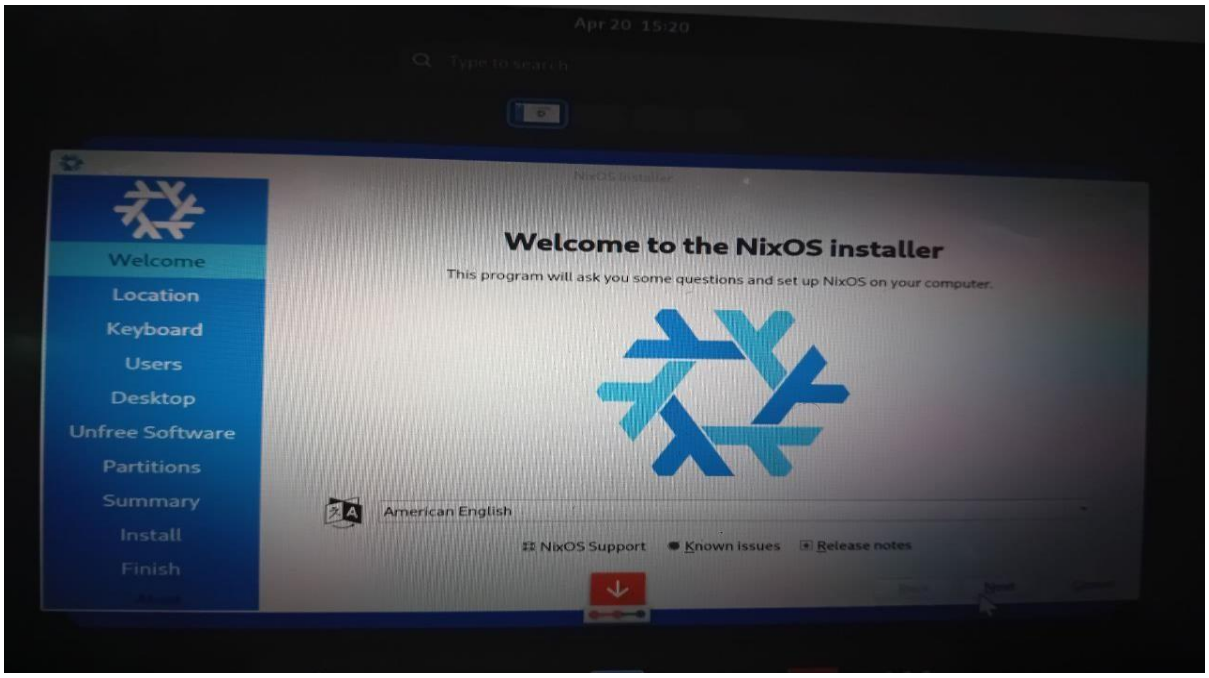
7. Shutdown

- This simply shuts down your system.
- Use this if you want to exit without installing.

In my case, I chose the first one (the default NixOS Installer) which is the most common and recommended way to start with.


NixOS [Running] - Oracle VirtualBox
File Machine View Input Devices Help
NixOS
NixOS 24.11.716947.26d499fc9f1d Installer
NixOS 24.11.716947.26d499fc9f1d Installer (nomodeset)
NixOS 24.11.716947.26d499fc9f1d Installer (copytoram)
NixOS 24.11.716947.26d499fc9f1d Installer (debug)
HiDPI, Quirks and Accessibility
Firmware Setup
Shutdown
Auto capture keyboard
Don't show again

❖ Afterwards, the welcome page will be displayed.



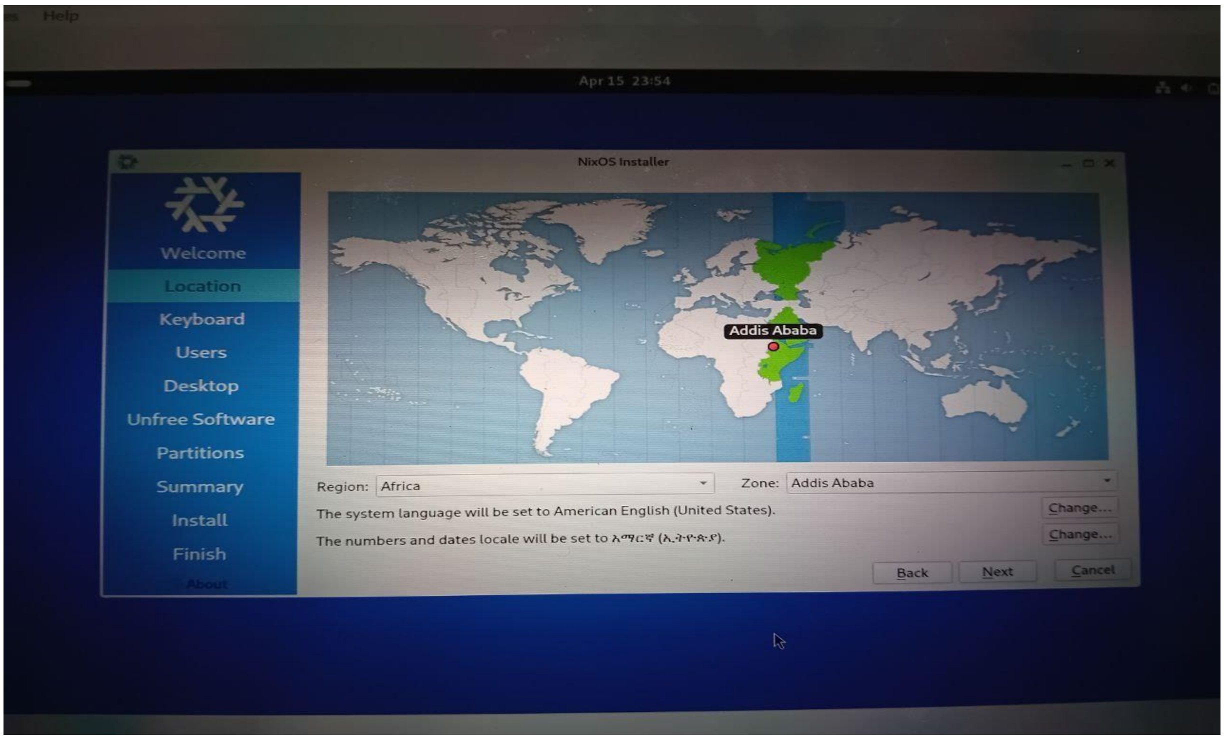
Welcome to the NixOS installer

This program will ask you some questions and set up NixOS on your computer.

American English

NixOS Support · Known issues · Release notes

❖ Click next, Location page will be presented. Choose the region and zone of yours. Then click next.

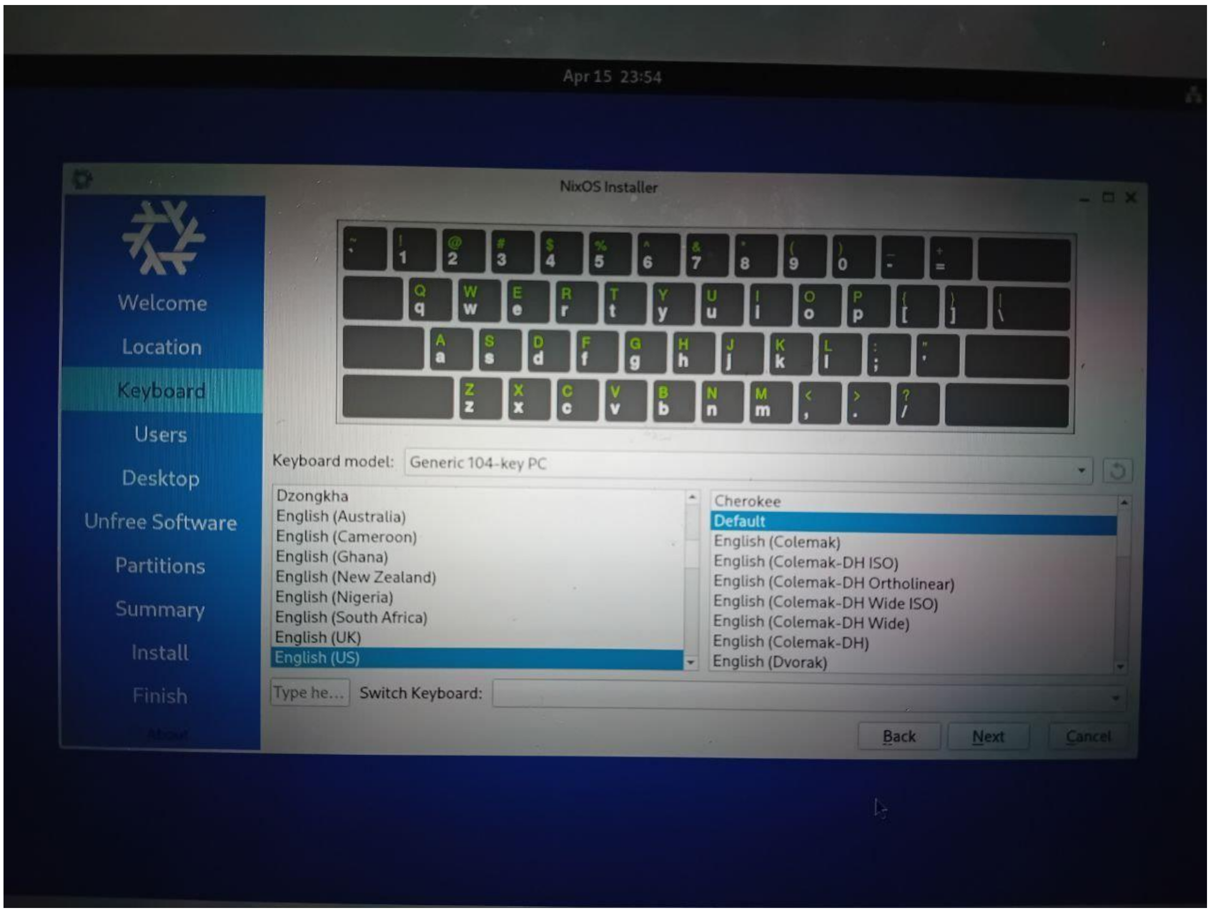


Region: Africa Zone: Addis Ababa

The system language will be set to American English (United States).

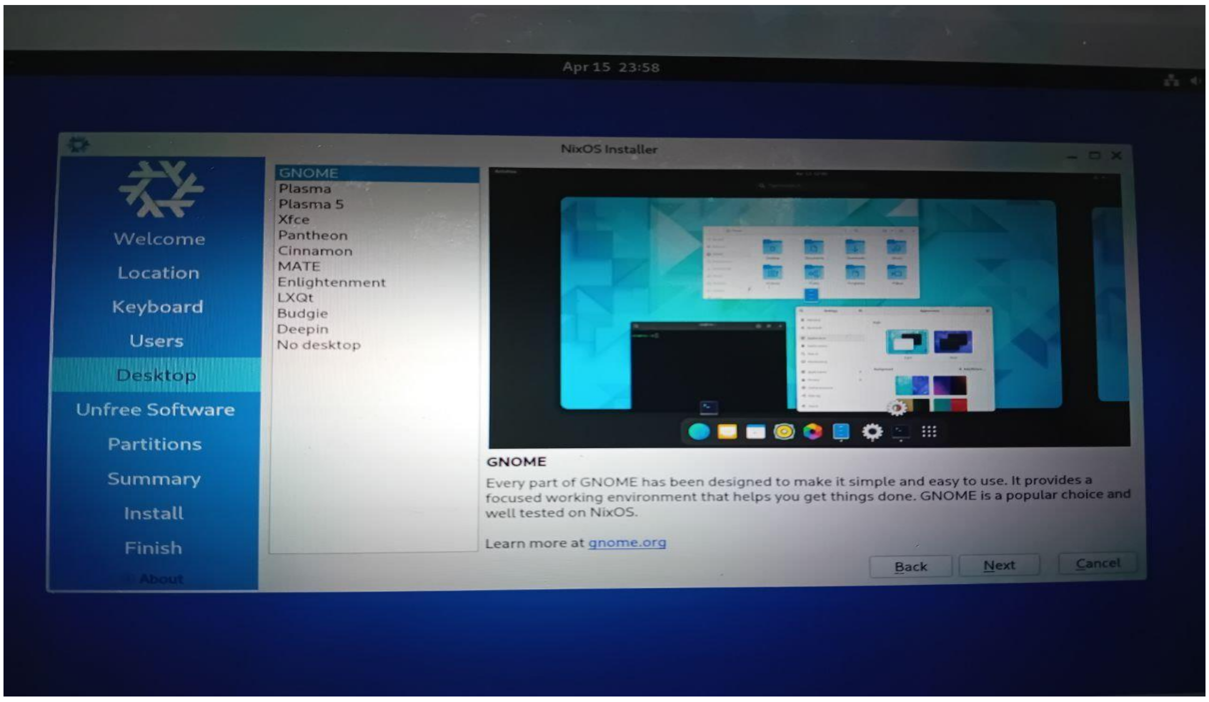
The numbers and dates locale will be set to አማርኛ (ኢትዮጵያ).

❖ Select the keyboard type, language and click next. I chose as you can see below.



Keyboard model: Generic 104-key PC

❖ Pick the desktop environment you want from the options. I go for GNOME.

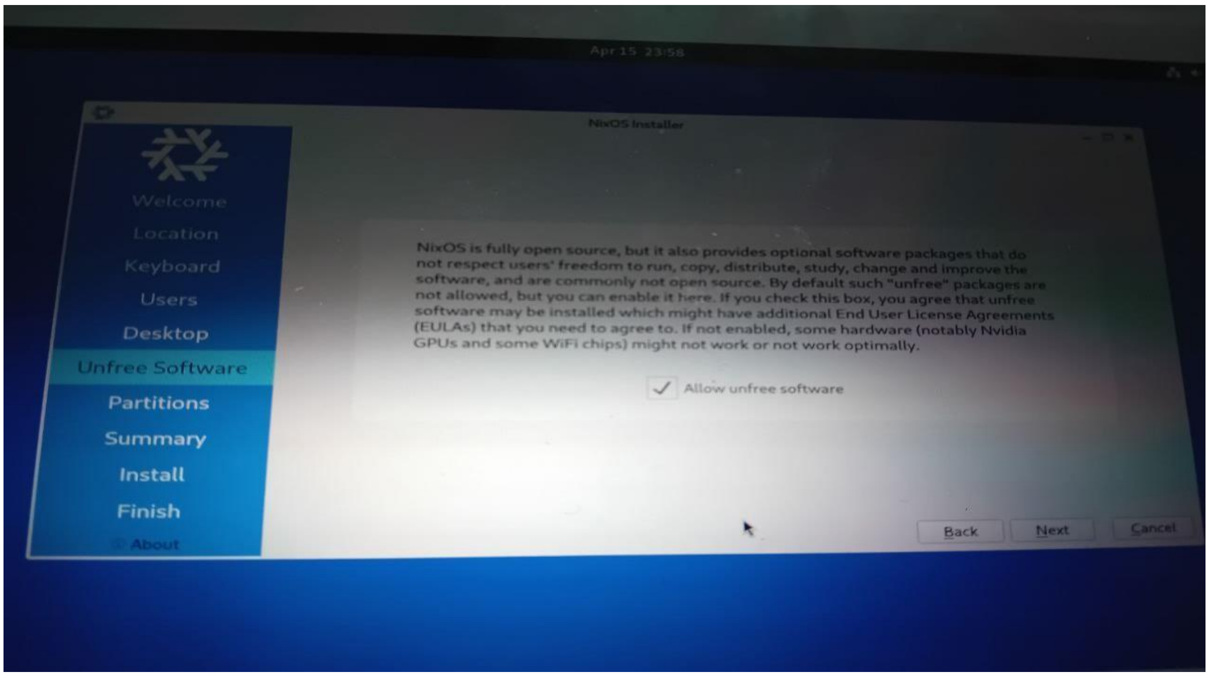


GNOME
Every part of GNOME has been designed to make it simple and easy to use. It provides a focused working environment that helps you get things done. GNOME is a popular choice and well tested on NixOS.

Learn more at gnome.org

❖ In this page it will ask you to "Allow unfree software". Although NixOS comes with free and open-source software as the default, I chose to permit unfree software for

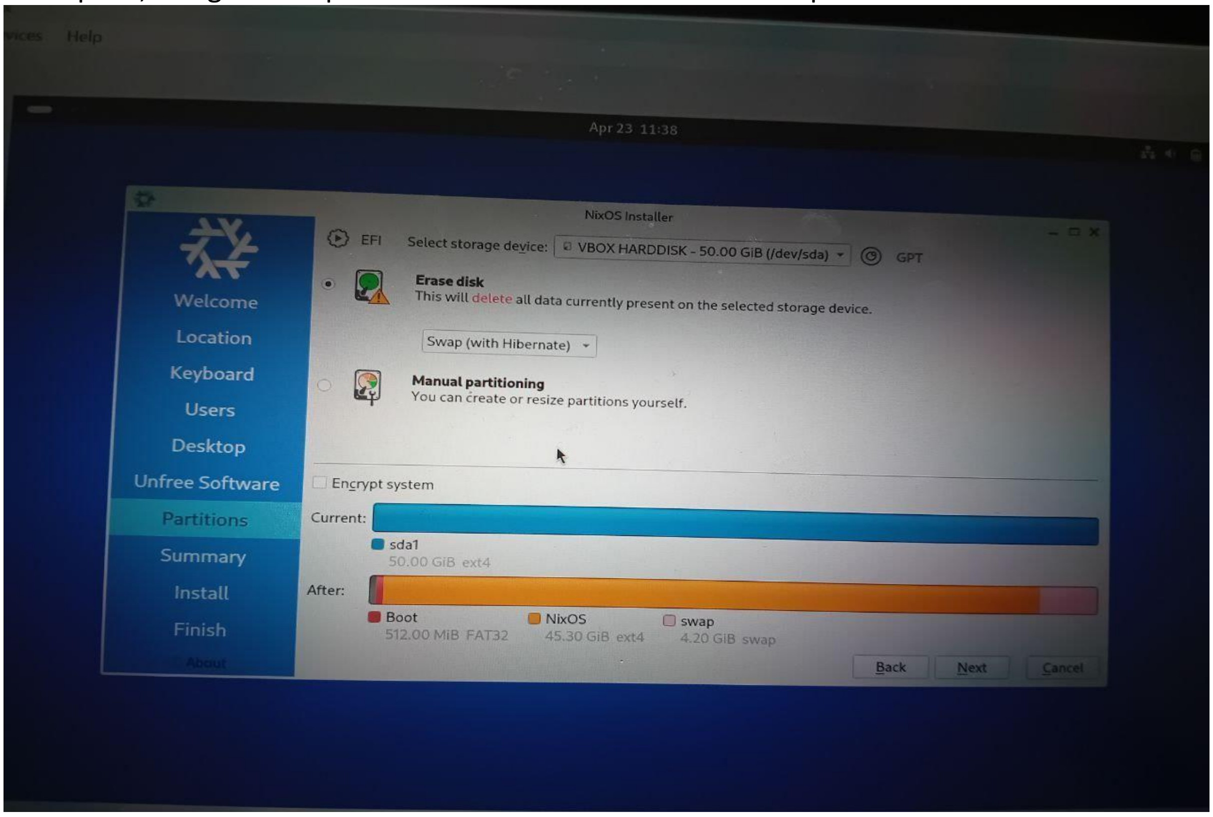
the following reasons:
1. Hardware Compatibility
There are many important drivers and firmware (specifically for Wi-Fi, graphic, audio) that are available as unfree packages only.
If I had not permitted unfree software, my system might not work properly or recognize all hardware devices — especially in a VirtualBox environment.
2. Required Access to Productive Proprietary Tools
Some pieces of software and tools used daily (like Google Chrome, Microsoft fonts, Zoom, or VirtualBox Extension Pack) are unfree but require access for productivity and interactivity in daily situations.
3. Installation Requirements of Practice
Some graphically or network-oriented bits (like GNOME extra drivers or VirtualBox additions) might silently need unfree packages upon installation. Without them Installation may fail or some specific functionality might fail after the installation.
And click next


NixOS Installer

NixOS is fully open source, but it also provides optional software packages that do not respect users' freedom to run, copy, distribute, study, change and improve the software, and are commonly not open source. By default such "unfree" packages are not allowed, but you can enable it here. If you check this box, you agree that unfree software may be installed which might have additional End User License Agreements (EULAs) that you need to agree to. If not enabled, some hardware (notably Nvidia GPUs and some WiFi chips) might not work or not work optimally.

✓ Allow unfree software

- In the partition page, pick out erase disk and swap with hibernate and click next . I used the Graphical Installer and selected "Erase Disk" to start with a clean installation and avoid partition problems. Also, "Swap with Hibernate" to automatically set up the swap partition and avoided swap errors I had experienced before. Even though I previously installed NixOS successfully after manually creating a swap file, using these options streamlined and stabilized the process.



EFI Select storage device: VBOX HARDDISK - 50.00 GiB (/dev/sda) GPT

(●) **Erase disk**
This will delete all data currently present on the selected storage device.

Swap (with Hibernate)

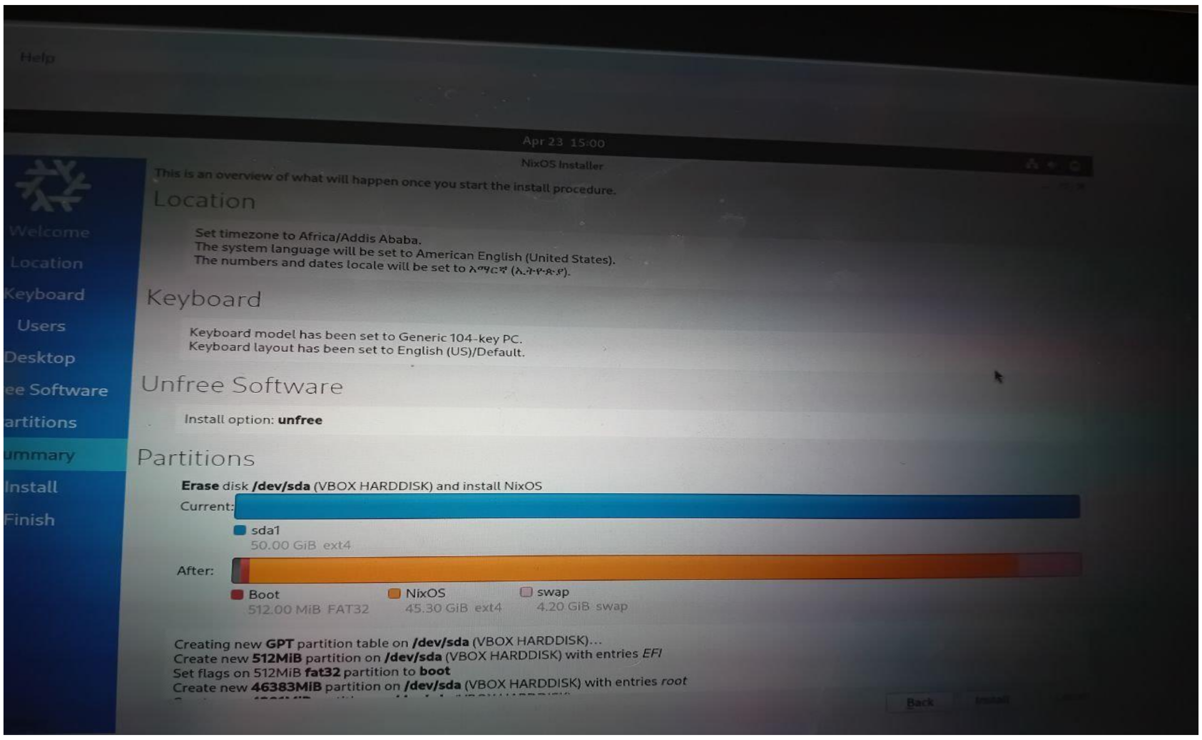
( ) **Manual partitioning**
You can create or resize partitions yourself.

☐ Encrypt system

Current: sda1 50.00 GiB ext4

After: Boot 512.00 MiB FAT32 | NixOS 45.30 GiB ext4 | swap 4.20 GiB swap

❖ After all steps we are now in summary which will show you what you have done so far. Click install and it’ll install it.



This is an overview of what will happen once you start the install procedure.

Location

Set timezone to Africa/Addis Ababa.
The system language will be set to American English (United States).
The numbers and dates locale will be set to አማርኛ (ኢትዮጵያ).

Keyboard

Keyboard model has been set to Generic 104-key PC.
Keyboard layout has been set to English (US)/Default.

Unfree Software

Install option: **unfree**

Partitions

Erase disk **/dev/sda** (VBOX HARDDISK) and install NixOS

Current: sda1 50.00 GiB ext4

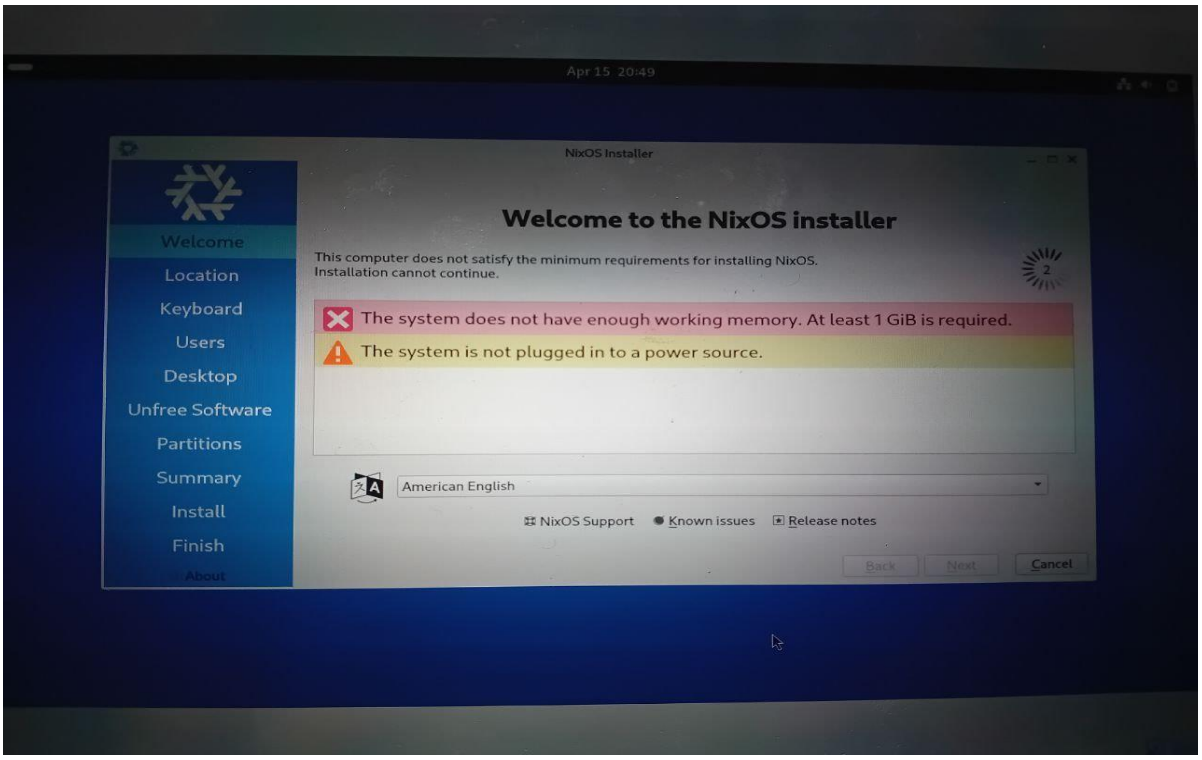
After: Boot 512.00 MiB FAT32 | NixOS 45.30 GiB ext4 | swap 4.20 GiB swap

Creating new **GPT** partition table on **/dev/sda** (VBOX HARDDISK)...
Create new **512MiB** partition on **/dev/sda** (VBOX HARDDISK) with entries *EFI*
Set flags on 512MiB **fat32** partition to **boot**
Create new **46383MiB** partition on **/dev/sda** (VBOX HARDDISK) with entries *root*

❖ After installing is done you will restart it and open the login page.

4) Issues (problems faced)

- Because of the problem below, I can’t install the os.



Apr 15 20:49

NixOS Installer

Welcome to the NixOS installer

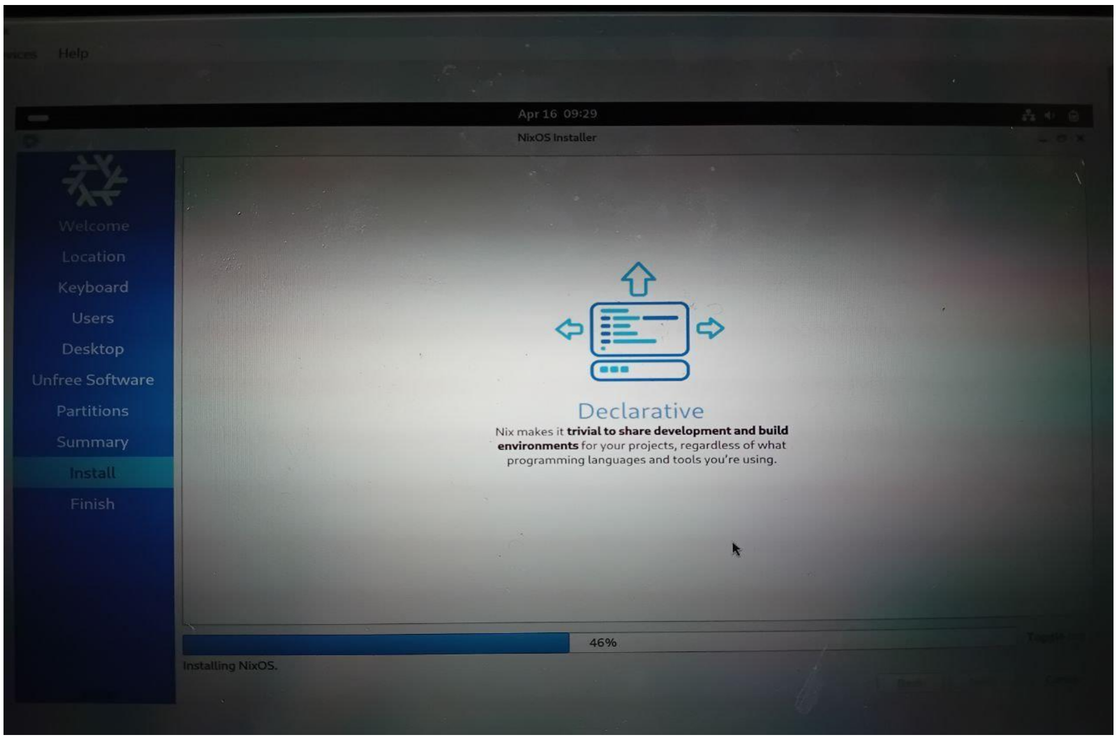
This computer does not satisfy the minimum requirements for installing NixOS.
Installation cannot continue.

The system does not have enough working memory. At least 1 GiB is required.

The system is not plugged in to a power source.

American English

- Stuck at 46% during installation.



Apr 16 09:29

NixOS Installer

Declarative

Nix makes it **trivial to share development and build environments** for your projects, regardless of what programming languages and tools you're using.

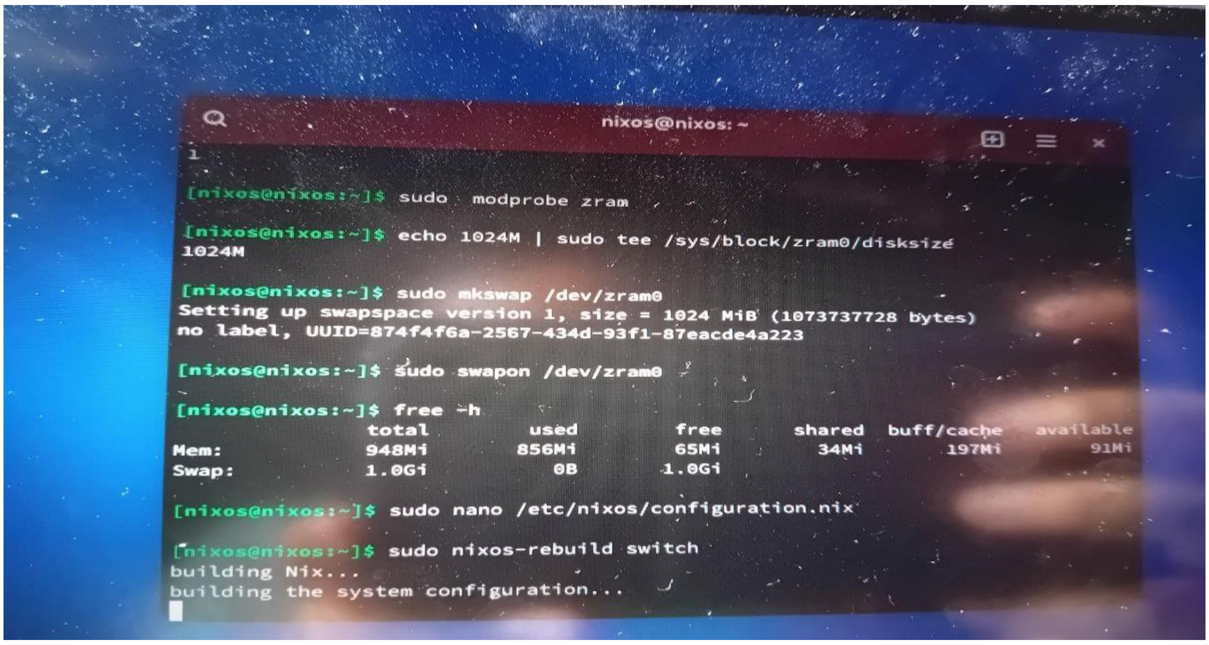
46%

Installing NixOS.

❖ The installer failed to create partition table at VBOX harddisk.

5) Solution

❖ For the first image problem 1 “system doesn’t have enough working memory” , I tried to solve it using bash.

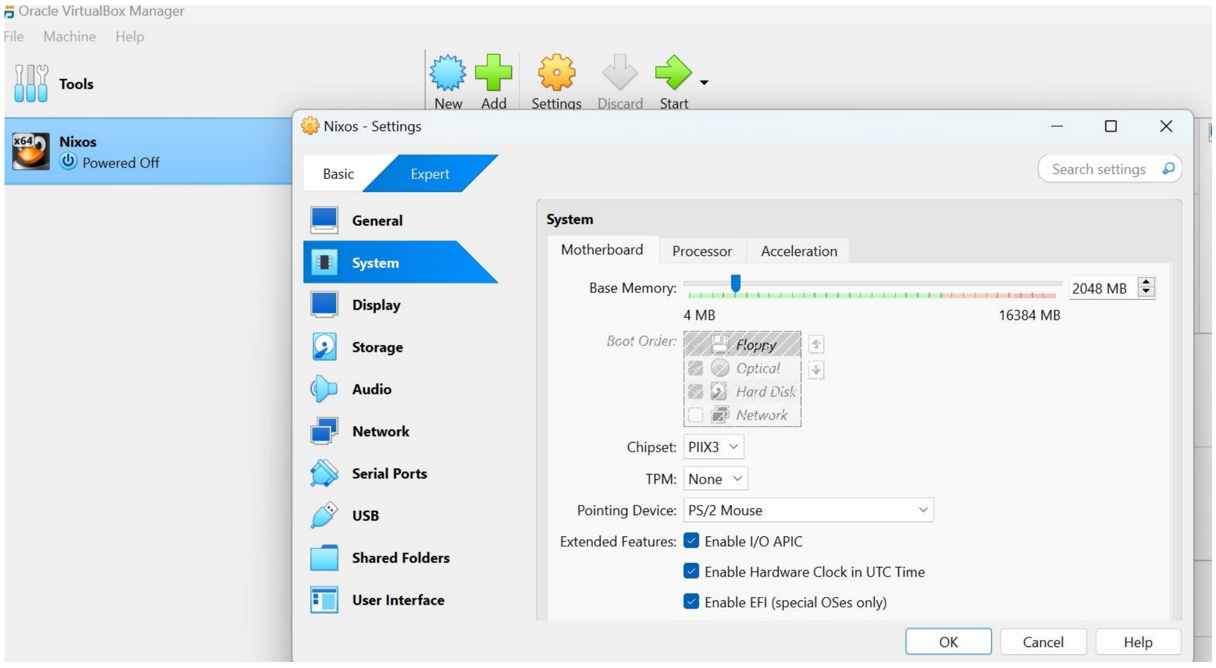


```
nixos@nixos: ~
1

[nixos@nixos:~]$ sudo  modprobe zram

[nixos@nixos:~]$ echo 1024M | sudo tee /sys/block/zram0/disksize
1024M

[nixos@nixos:~]$ sudo mkswap /dev/zram0
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=874f4f6a-2567-434d-93f1-87eacde4a223

[nixos@nixos:~]$ sudo swapon /dev/zram0

[nixos@nixos:~]$ free -h
               total        used        free      shared  buff/cache   available
Mem:           948Mi       856Mi        65Mi        34Mi       197Mi        91Mi
Swap:          1.0Gi          0B       1.0Gi

[nixos@nixos:~]$ sudo nano /etc/nixos/configuration.nix

[nixos@nixos:~]$ sudo nixos-rebuild switch
building Nix...
building the system configuration...
```

- sudo modprobe zram= Loads the ZRAM module.

- echo 1024M | sudo tee /sys/block/zram0/disksize= Sets up 1024 MiB ZRAM size.
- sudo mkswap /dev/zram0= Initializes the ZRAM device as swap.
- sudo swapon /dev/zram0= Enables it as swap.
- free -h = shows Swap: 1.0Gi total, 0B used – confirms swap is active and unused yet.
- sudo nano /etc/nixos/configuration.nix= make ZRAM swap persistent.Then add the below statement after nano /etc/nixos/configuration.nix opened
- {
    - boot.initrd.kernelModules = ["zram"];
    - boot.zramSwap.enable = true;
    - boot.zramSwap.memoryPercent = 50; # or set size explicitly
    - }
- sudo nixos-rebuild switch= Applies the changes.

Even if I add one GB using bash, it doesn’t avoid the problem. So, I changed its base memory to 2048 manually. shut down the virtual machine. It cannot be adjusted while the VM is in a saved or running state> Open VirtualBox.>Select your NixOS VM from the list.>Click on "Settings".>Go to the "System" tab.>Under the "Motherboard" tab, you’ll see "Base Memory".>Drag the slider or enter a new value e.g., increase to 2048 MB>Click OK to save.



System

Motherboard | Processor | Acceleration

Base Memory: 2048 MB (4 MB – 16384 MB)

Boot Order: Floppy, Optical, Hard Disk, Network

Chipset: PIIX3

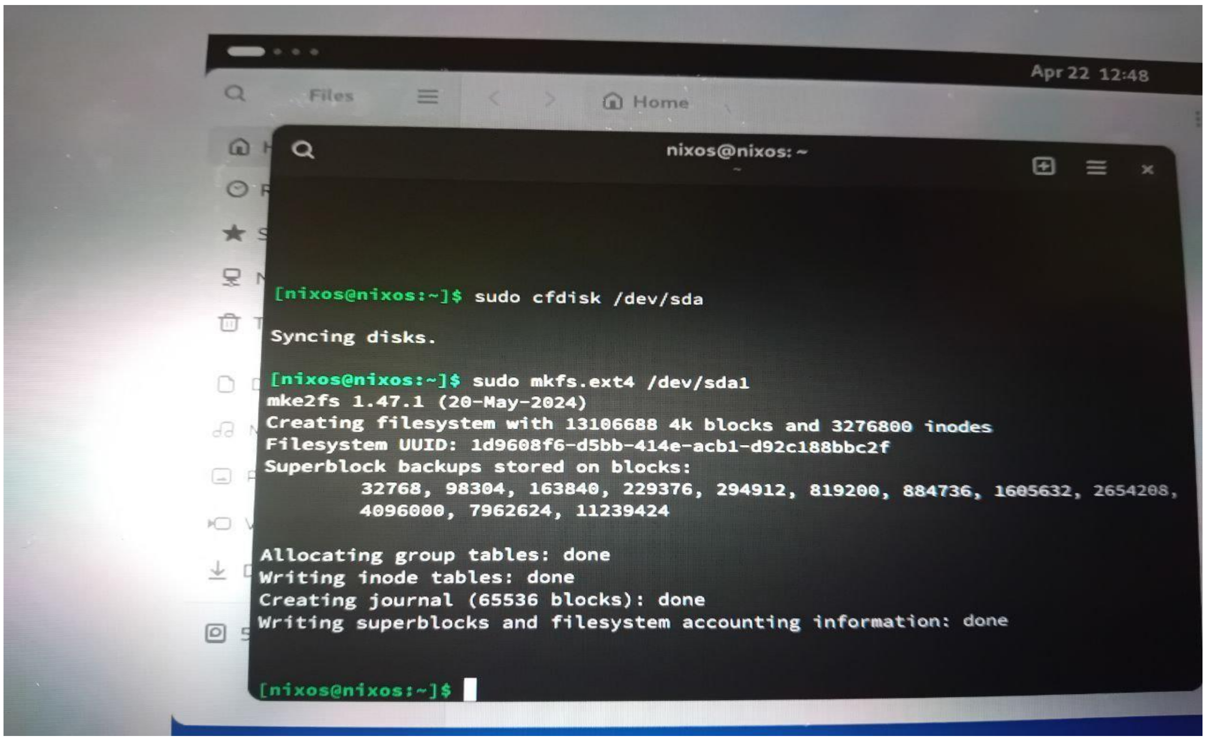
TPM: None

Pointing Device: PS/2 Mouse

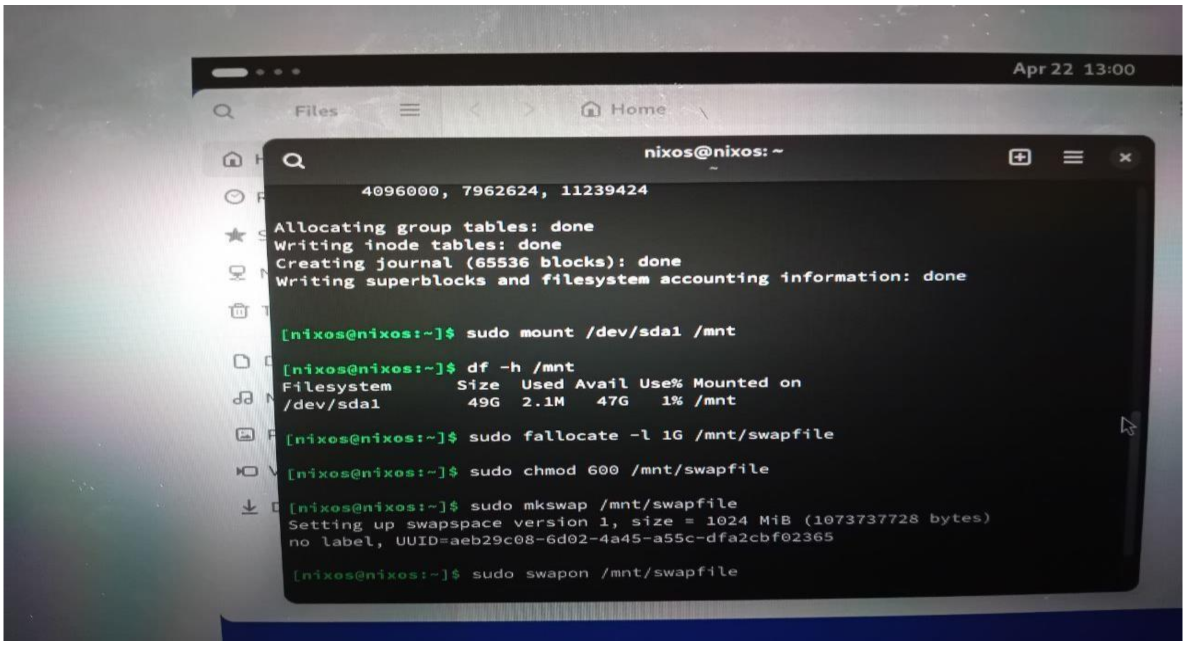
Extended Features: ✓ Enable I/O APIC
✓ Enable Hardware Clock in UTC Time
✓ Enable EFI (special OSes only)

Problem 2 which says "The system is not plugged into a power source." This problem appears whenever your pc is not connected to its charger. After plugging it to its charger it will be erased.

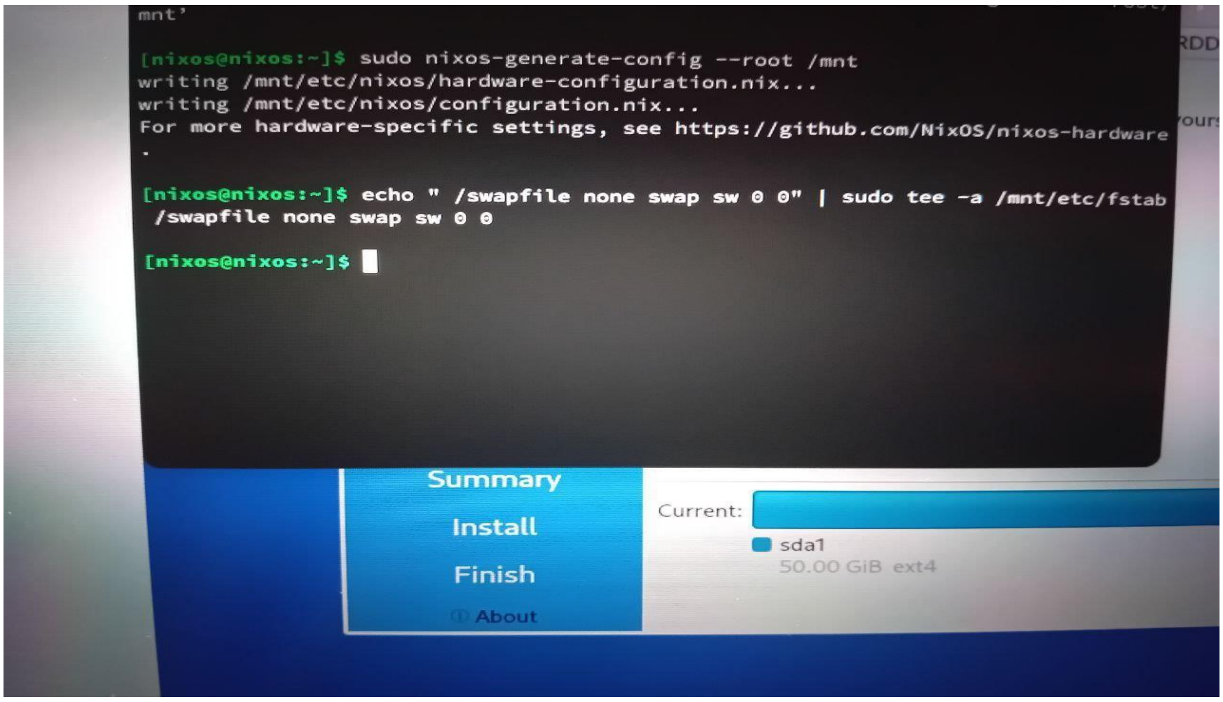
❖ During installation of NixOS, the installation would be stuck at 46%, and it would not continue. This is usually caused by:
The system does not have enough memory (especially in VirtualBox with minimal RAM).
There is not enough swap space, and hence the installer cannot load everything it needs.
The partition configuration or disk space allocation may be wrong or inadequate.
To fix the problem, I followed these steps:
1. Deleted all existing partitions using cfdisk or manual partitioning.
2. Created a new primary partition that used the full disk space.
3. Manually allocated swap in /mnt



```
[nixos@nixos:~]$ sudo cfdisk /dev/sda

Syncing disks.

[nixos@nixos:~]$ sudo mkfs.ext4 /dev/sda1
mke2fs 1.47.1 (20-May-2024)
Creating filesystem with 13106688 4k blocks and 3276800 inodes
Filesystem UUID: 1d9608f6-d5bb-414e-acb1-d92c188bbc2f
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424

Allocating group tables: done
Writing inode tables: done
Creating journal (65536 blocks): done
Writing superblocks and filesystem accounting information: done

[nixos@nixos:~]$ 
```

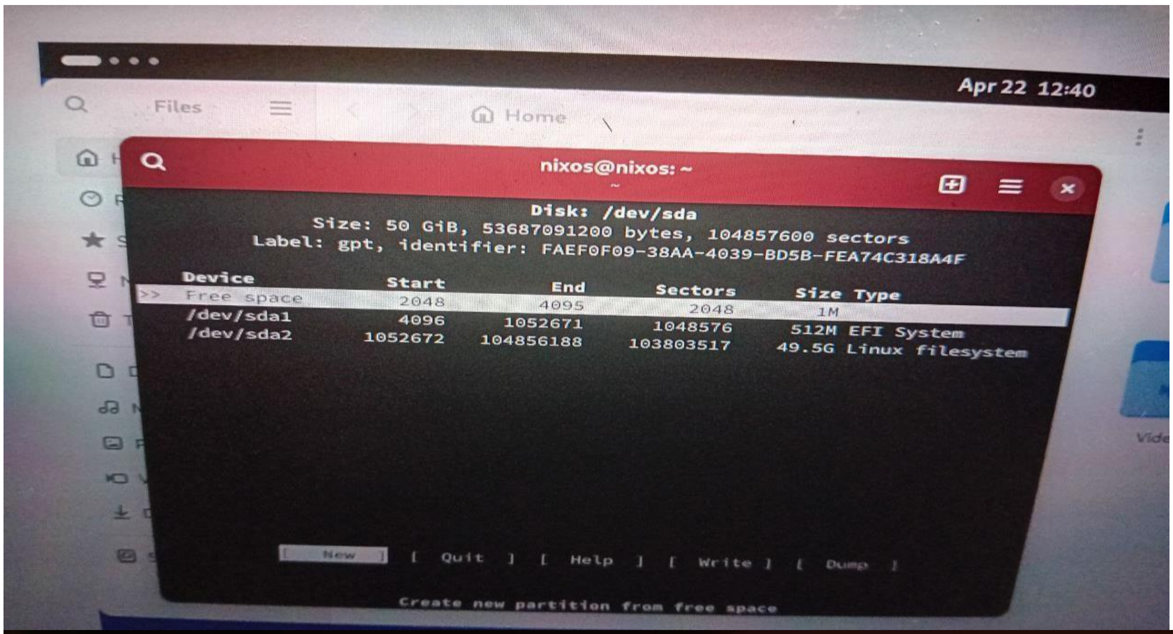


```
        4096000, 7962624, 11239424

Allocating group tables: done
Writing inode tables: done
Creating journal (65536 blocks): done
Writing superblocks and filesystem accounting information: done


[nixos@nixos:~]$ sudo mount /dev/sda1 /mnt

[nixos@nixos:~]$ df -h /mnt
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        49G  2.1M   47G   1% /mnt

[nixos@nixos:~]$ sudo fallocate -l 1G /mnt/swapfile

[nixos@nixos:~]$ sudo chmod 600 /mnt/swapfile

[nixos@nixos:~]$ sudo mkswap /mnt/swapfile
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=aeb29c08-6d02-4a45-a55c-dfa2cbf02365

[nixos@nixos:~]$ sudo swapon /mnt/swapfile
```

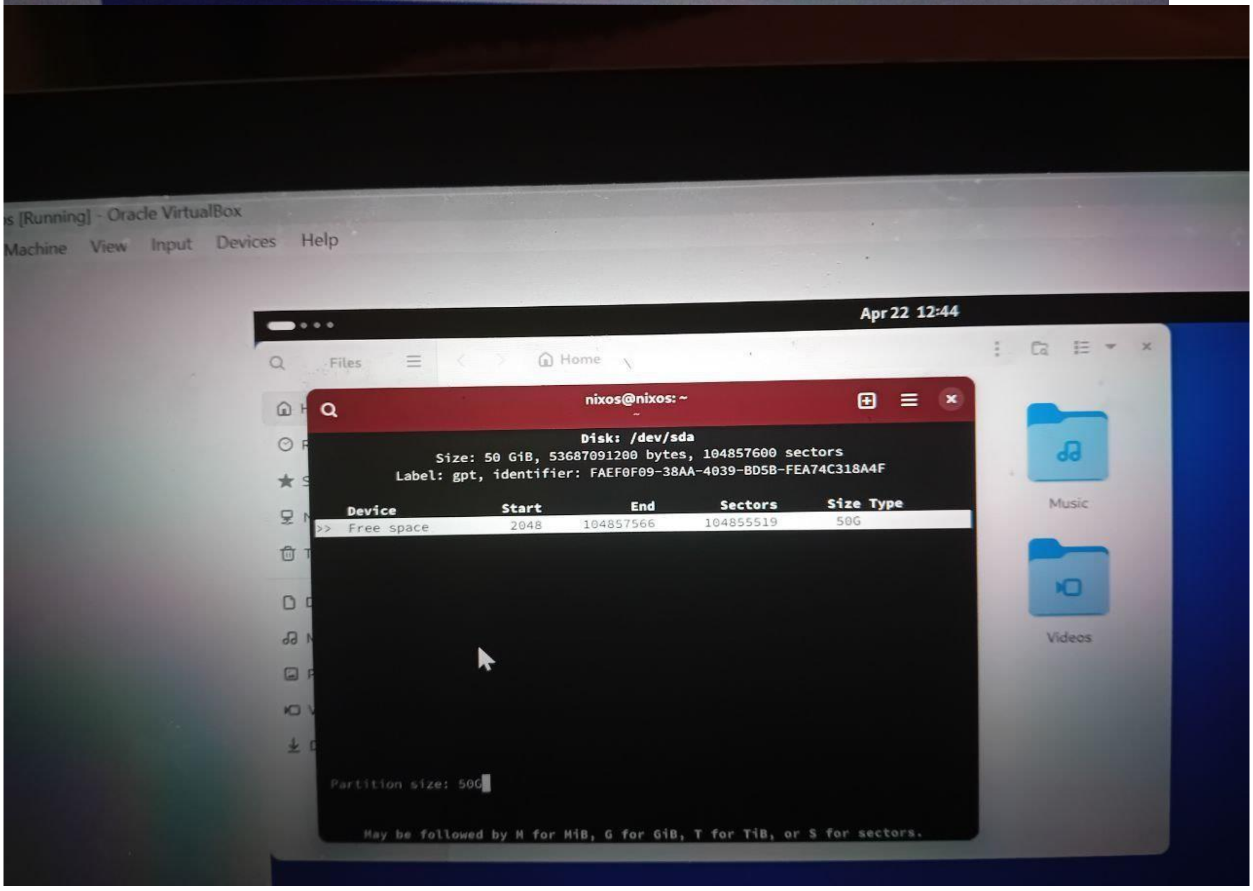


```
mnt'

[nixos@nixos:~]$ sudo nixos-generate-config --root /mnt
writing /mnt/etc/nixos/hardware-configuration.nix...
writing /mnt/etc/nixos/configuration.nix...
For more hardware-specific settings, see https://github.com/NixOS/nixos-hardware
.

[nixos@nixos:~]$ echo " /swapfile none swap sw 0 0" | sudo tee -a /mnt/etc/fstab
 /swapfile none swap sw 0 0

[nixos@nixos:~]$
```

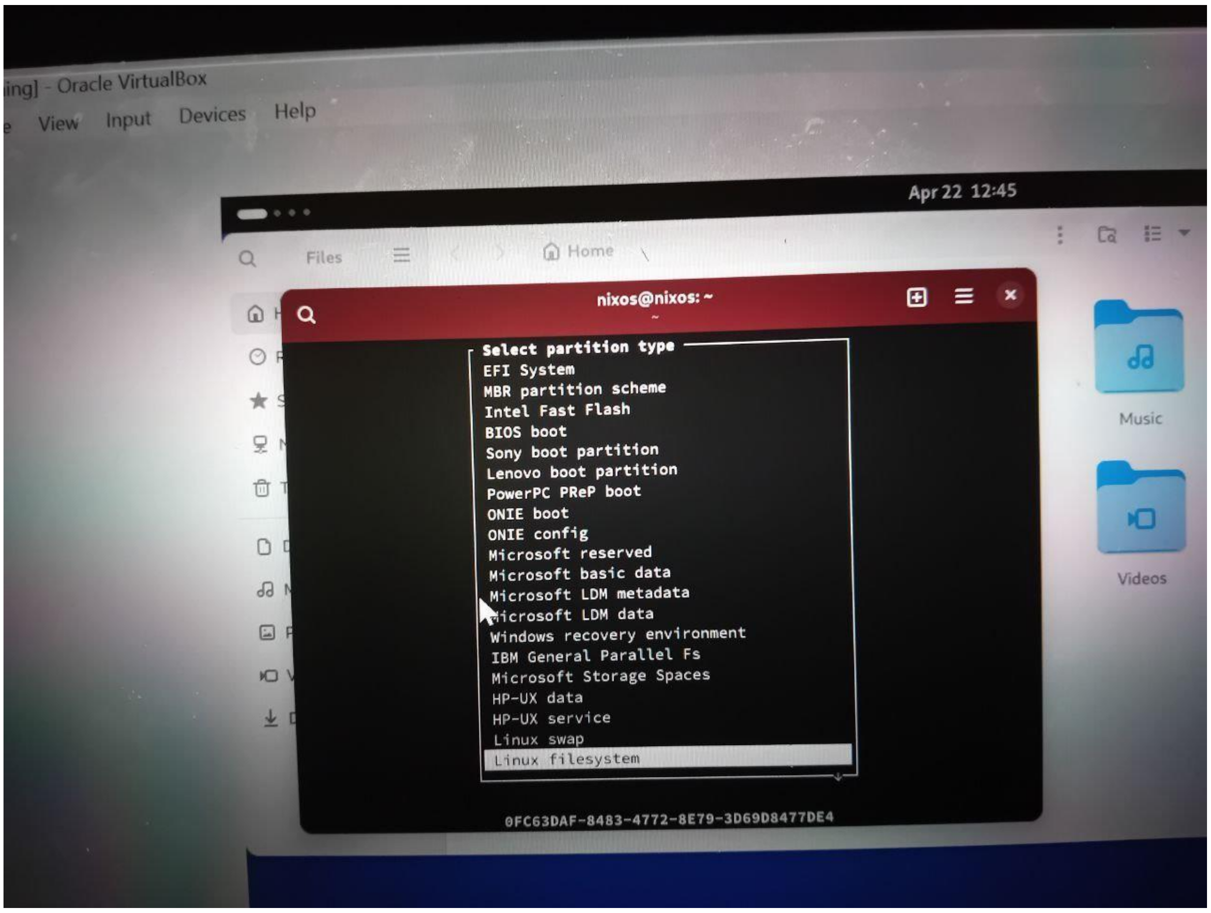
- ✓ After running sudo cfdisk /dev/sda which Opens the Partitioning Tool:
- ✓ To delete Pre-existing Partitions, Go through each existing partition in cfdisk and delete them one by one.You should now have "Free space" occupying the whole disk. Then, create a New Primary Partition, click on "New"> Click on Primary> Use all disk space available for this partition > Set the type if needed (e.g., Linux filesystem) > Choose "Write", enter "yes" to accept, and exit cfdisk.
- ✓ sudo mkfs.ext4 /dev/sda1= This command formats a partition with the ext4 filesystem which has become standard for Linux. It sets the disk file system to be used for installing files on it.
- ✓ sudo mount /dev/sda1 /mnt= That mounts the newly formatted partition automatically at /mnt. This is the point where the installer would root the new NixOS.
- ✓ sudo fallocate -l 2G /mnt/swapfile= This will create a 2GB swap file in the mounted partition.
- ✓ sudo chmod 600 /mnt/swapfile= Secures the swap file so that only the system can access it.
- ✓ sudo mkswap /mnt/swapfile= Formats the file as swap space.
- ✓ echo "/swapfile none swap sw 0 0" | sudo tee -a /mnt/etc/fstab= Adds the swap file to the system's boot configuration file (fstab) so it is enabled automatically on every boot.
- ✓ sudo nixos-generate-config --root /mnt=This command generates the default NixOS configuration files based on the system. This creates: configuration.nix - holds settings regarding system configuration .hardware-configuration.nix - it notices hardware and partitions . It takes /mnt as the root because that is where the target system has been mounted.



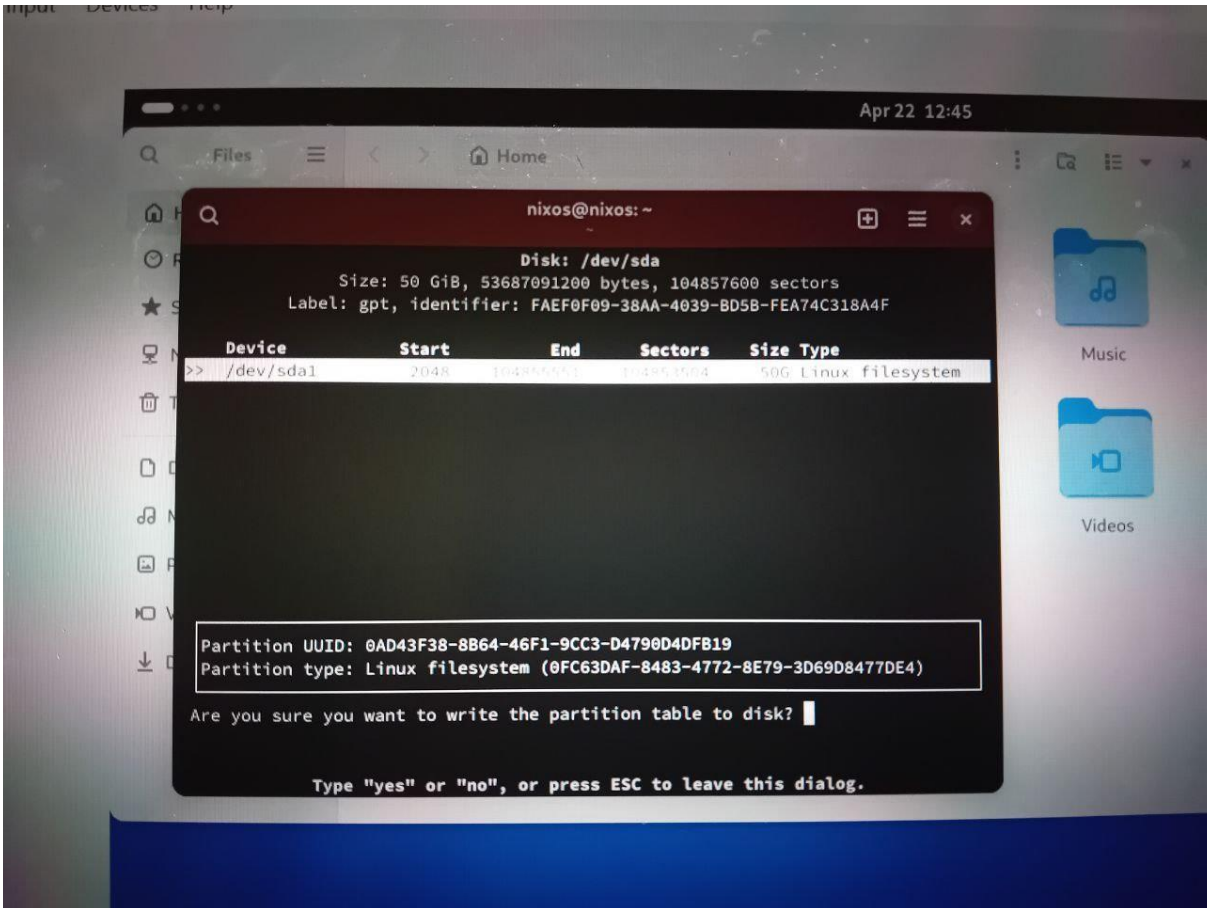
```
                        Disk: /dev/sda
        Size: 50 GiB, 53687091200 bytes, 104857600 sectors
      Label: gpt, identifier: FAEF0F09-38AA-4039-BD5B-FEA74C318A4F

    Device              Start         End      Sectors    Size Type
>>  Free space           2048        4095         2048      1M
    /dev/sda1            4096     1052671      1048576    512M EFI System
    /dev/sda2         1052672   104856188    103803517   49.5G Linux filesystem

         [  New  ]  [  Quit  ]  [  Help  ]  [  Write  ]  [  Dump  ]

                   Create new partition from free space
```


s [Running] - Oracle VirtualBox
Machine View Input Devices Help
Apr 22 12:44
Files
Home
nixos@nixos: ~
Music
Videos

```
                        Disk: /dev/sda
        Size: 50 GiB, 53687091200 bytes, 104857600 sectors
      Label: gpt, identifier: FAEF0F09-38AA-4039-BD5B-FEA74C318A4F

    Device              Start         End      Sectors    Size Type
>>  Free space           2048   104857566    104855519     50G




 Partition size: 50G

     May be followed by M for MiB, G for GiB, T for TiB, or S for sectors.
```



```
Select partition type
EFI System
MBR partition scheme
Intel Fast Flash
BIOS boot
Sony boot partition
Lenovo boot partition
PowerPC PReP boot
ONIE boot
ONIE config
Microsoft reserved
Microsoft basic data
Microsoft LDM metadata
Microsoft LDM data
Windows recovery environment
IBM General Parallel Fs
Microsoft Storage Spaces
HP-UX data
HP-UX service
Linux swap
Linux filesystem

0FC63DAF-8483-4772-8E79-3D69D8477DE4
```



```
                     Disk: /dev/sda
       Size: 50 GiB, 53687091200 bytes, 104857600 sectors
    Label: gpt, identifier: FAEF0F09-38AA-4039-BD5B-FEA74C318A4F

    Device          Start        End      Sectors   Size Type
>>  /dev/sda1        2048  104855551    104853504    50G Linux filesystem

 Partition UUID: 0AD43F38-8B64-46F1-9CC3-D4790D4DFB19
 Partition type: Linux filesystem (0FC63DAF-8483-4772-8E79-3D69D8477DE4)

Are you sure you want to write the partition table to disk?

          Type "yes" or "no", or press ESC to leave this dialog.
```

❖ And I tried to install it but the installation failed so Later, I used the "Erase Disk" and "Swap with Hibernate" features to enable the installer to automatically manage memory and disk space — which totally fixed the problem.

6) <u>File system support</u>

❖ NixOS has the most popular filesystems that you can choose from.

1. ext4 (Fourth Extended Filesystem) = This is the most popular Linux filesystem, good for almost any use. Fast and stabilizing, the usage does not need a big setup. Unless you have a set of specific needs, you're barely wrong in choosing ext4 or just plain smart.

2. Btrfs = It is more of an advanced file system with features like snapshots that enable rolling back your system after an imperfect state. It involves compression and sub volumes worthy of any tinkerer or anyone who requires full-fledged data management.

3. FAT32= It is an old filesystem that is implemented everywhere from Linux to Windows to game consoles. For NixOS, it is often called to duty for UEFI boot partitions. No use would you put it for your actual system; it does not support large files and is hardly stable over time.

4. exFAT= This one's like the smart cousin of FAT32. It can handle bigger files and is ideal for external disks or USBs that you use across different operating systems (Windows and macOS). NixOS supports it if you need sharing less of a headache.

5. NTFS= It is the native Windows filesystem. You'll want NTFS support in order to share your Windows files with NixOS while in a dual configuration. Not the best root filesystem for Linux, but fine for sharing hates and files.

6. ZFS (Zettabyte file system) = is high-end stuff, and an industrial-strength filesystem; serious workloads are what it's made for and comes with self-heal, snapshot, and deduplication. Can be used with NixOS; you will have to enable support manually. Great for servers and professionals with maximum reliability in mind.

7. HFS+ and APFS= These are Apple filesystems. NixOS can mount drives formatted this way but the support is limited, generally read-only. Handy for moving files off a Mac but not something you'd use as a main filesystem on NixOS.

❖ I pick out ext4.here are my reasons why I chose it:

✓ Consistency and dependability= The extended filesystem (ext) has existed for a long time and is now in its fourth iteration, or ext4. Because of its exceptional stability and dependability over time, it continues to be the default option for the majority of Linux distributions, including NixOS. It has been thoroughly tested in a variety of settings, ranging from large servers to personal workstations, and that dependability is precisely what I need when configuring a system.

- ✓ easy Setup= I required a filesystem that was simple to install and did not require sophisticated technical knowledge or special setup. ext4 is simple to format and mount. In contrast to more complex filesystems like Btrfs or ZFS, ext4 does not require additional configuration options or tools for basic use. This simplicity made it a simple choice for me, as I can focus on other parts of the NixOS installation rather than getting bogged down in filesystem problems.
- ✓ Low Risk= I'm new to NixOS and Linux, and I wanted a filesystem that wouldn't be losing data and cause trouble later on. ext4 is not as shiny as some other filesystems, like snapshots, but it is simple and has been around long enough to make me rest easy. There is no configuration error and surprising filesystem behavior that sometimes happens with more advanced filesystems.
- ✓ First-rate in any general sense, ext4 performance is perfectly sufficient for most use cases. It does lack the advanced features offered by modern filesystems like Btrfs (compression, snapshots) or ZFS (data integrity, self-healing), but is very good for day-to-day work. Ext4 is the default choice for the typical Linux installation, as it supplies just the right measure of speed and stability, meaning I won't have usually bad experiences with slowdowns or system instability while just working on the system.
- ✓ Support Released for the Long term= Given its reputation as a filesystem after years of development and extensive use under different distributed environments, one could say that ext4 is potentially future proof. I understand that ext4 is not only a quality filesystem now but also one that would continue to receive periodic updates and improvements guided by the Linux community.
- ✓ Extensive Compatibility= Another reason for which I chose ext4 is because basically it is compatible with almost every Linux-based tool and application. It's default in most Linux distributions which imply most software is aimed toward using ext4. Therefore, in case I need to access data on this partition, from another machine with Linux or troubleshoot an issue, there won't be any compatibility problems. Other filesystems like Btrfs or ZFS may require some additional setup or modules which would pose a challenge if I were required to troubleshoot or transfer data between systems.

7) Advantages and Disadvantages

❖ Advantages

- ▪ Declarative Configuration= NixOS is built around a very powerful concept: the declarative configuration of software packages. Instead of installing and configuring software manually, you simply define your system's desired state in a single file: configuration.nix. This file takes care of everything from the installation of packages to system settings and services. The beauty of this approach is that it allows you to recreate or replicate your environment on multiple machines without having to configure them all by hand. Devs, sysadmins, and all other "people having to have the same conditions across their machines" find this to be a real game changer.

- Atomic Upgrades & Rollbacks= One of the best features of NixOS is that it lets you do atomic system upgrades. NixOS defines that an upgrade should be treated as a single indivisible action; therefore, if something goes wrong in the middle of the process, you need not panic. You have the ability to very easily roll back to a previously working configuration. This further enhances the value of NixOS for anyone running mission-critical applications or systems where even a slight misfortune causes major problems.
- Customizable= Because of the Nix package manager, NixOS is extremely customizable. You are not limited to some predefined software repositories as in the case of normal Linux distributions; rather, almost every part of an OS and its associated software can be customized. This means that you can build a very precise configuration for your environment or use bleeding-edge software—whatever it is for you—that it be exact for 'your' needs: that is what you can achieve using NixOS.
- Reproducible Environments= Reproducibility is one of the core philosophies of NixOS, meaning that once configured, sytem 'configuration.nix' can be recreated on another machine with diminishedworking efforts. It is extremely useful in situations such as software development, wherein team members often have to make sure that every development machine, as well as a working production environment, are identical. Very useful in DevOps when implementing automated deployments and infrastructure provisioning.

❖ Disadvantages

- Steeper Learning Curve=NixOS's declarative approach provides many benefits but requires a learning curve. Most Linux distributions allow for more of a trial-and-error approach to configuration (ranging from editing files to installing different packages) to allow a beginner to really learn their system. NixOS uses a central configuration file for all changes and might seem very abstract or confusing to the novice for how to configure. It is almost sort of a complete mind shift for learning NixOS, an experience which some may find intimidating or frustrating if they come here after years of familiarity with various Linux distributions.
- Less Software Repositories=There may not be a default package on NixOS, although NixOS has a repository of vast software packages. NixOS packages are built following an atypical manner through the Nix package manager; hence some software is commonly available in other distributions, like APT or YUM-based systems, that may not be available. Some software may also entail extra labor on the part of the users to build or wrap for it under NixOS. And believe me, this can really frustrate those users accustomed to the easy software access that many other easier systems provide.

- Requires a NixOS Mindset= NixOS forces users to think of a different way about declarative configurations instead of following other distributions that follow the usual imperative way. In other traditional systems, you would use apt or yum to install packages and configure the services on your own by manually editing the configuration files. On the contrary, everything from services to users and package installations has to be declared in configuration.nix in NixOS. For those people who are used to the traditional package managers and interactive system configuration, this way of inputting information may require a mindset transformation.

8) Conclusion

- Experiencing NixOS was new to me. Unlike most Linux distributions, that do have an installer, NixOS does not have one. It requires manual procedures of creating and formatting partitions, mounting them, and with the help of a configuration file aboard, managing the installation system setup. In the beginning, it was tough, but I tried to work on it by taking it gradually, and after a while, I was able to get the job done.I decided to go with ext4 as it provides stability, simplicity, and wide support. Btrfs or ZFS only complicate matters further in my opinion, since I am pretty much learning, with ext4 being the gentlest option that worked and with no extra tinkering involved.An odd thing about NixOS is the system management through a file called configuration.nix. That prevents the user from performing everything manually, thus compromising the whole system while editing here and there. It can use a bit of logic while describing how the system should be. It is a painstaking process to learn; I am still learning to get hang of it, but I do see the reason why it is supposed to be helpful- for edits, reinstallations, or repairs without starting afresh.While I'm not going to say that I understand all there is to know about NixOS, this installation process gave me a hint at its power and flexibility. Certainly not the easiest distro for beginners, but they learn a lot as they go. In the longer run, NixOS is more organized and trustworthy for people wishing for a system under full control and customization.I had a good experience overall. I still have much to learn, but NixOS has given me a chance to try something different and learn the basics of building Linux systems in a more structured way.

9) Future Outlook / Recommendations

- Boost Adoption in DevOps and Servers: The declarative configuration of NixOS makes it especially suitable for environments that call for automation and infrastructure as code. With increasing DevOps activities in enterprises, the reproducibility approach for systems configuration will gain more popularity for NixOS. It will be really beneficial for server environments, as the configuration of systems and software need to be the same on all these different machines to circumvent inconsistencies.

- Upgrade Documentation and Support: Documentation is one area in which NixOS could certainly improve. There is a lot of valuable information available from the NixOS community, but it can still be tough for beginners to get started. As the community grows, it is quite likely that user-friendly documentation, tutorials, and community support will also become stronger for NixOS. This decline in the barrier for new entrants to the ecosystem will serve to enhance the onboarding experience.
- Wider Package Support: As NixOS advances, there is bound to be better support for packages in its repository. Increasingly, software vendors may be inclined to officially support NixOS; more and more users are adopting it, and the package management system keeps maturing. This means that NixOS is becoming more and more relevant for common use, where users will have access to an abundance of prebuilt software.
- Niche To Mainstream: While NixOS is a niche operating system at present, there is indeed potential for mainstreams adoption in certain echelons, particularly for those who require consistency and automation. If NixOS can continue to mend its ecosystem and proliferate its user base, we would possibly see it being a much-spoken distribution of Linux in use by both persons and organizational systems.

10) Virtualization in Modern Operating Systems

- Virtualization defines the creating of a virtual habitation of a physical component such as a computer, storage, or network. In the operating system, it refers specifically to the running of multiple virtual machines (VMs) on a single physical machine, each of which behaves like its own independent computer with its accompanying operating system and applications, even though it is really running on the shared physical hardware._There is a plethora of reasons that command a larger implementation of virtualization in modern computing:

    - Resource Efficiency: Not withstanding using multiple physical machines for different tasks, an alternative is to standardize a machine with many VMs. It ameliorates cost, space, and energy consumption.
    - Testing and Development: Developers can verify runs with different operating systems or configurations without ever needing separate, physical computers for each. It may also be useful to learn as, in my situation, installing and testing NixOS in a virtual environment.
    - Isolation and Security: Each VM is isolated from the others, so if something goes wrong—such as crashing or a virus—it doesn't affect the rest.
    - Snapshots and Rollbacks: Many virtualization technologies will allow taking a snapshot, that is, saving the current state of a VM and go back to it later; thus, allowing an environment for testing or experimentation with a minimal risk.
    - Scalability: Virtualization helps in enterprise environments to scale systems up or down as per demand

- The core component that enables virtualization is called a hypervisor. Hypervisors are a software layer that creates virtual machines and manages their functioning by allocating system resources (CPU, memory, disk, etc.) to each VM.Two types of hypervisors exist:

    - Type 1 (Bare-metal): Hypervisors which run directly on the physical hardware. Examples are VMware ESXi and Microsoft Hyper-V.
    - Type 2 (Hosted): Run on top of a host operating system, Examples are Oracle VirtualBox and VMware Workstation, mostly used for personal and educational purposes.

10.1) virtualization in Nixos

- Virtualization in NixOS means running another operating system in a virtual machine (VM), or using NixOS itself inside a VM. For me, I used VirtualBox to install NixOS as a virtual machine, so my main system stayed safe. It's like creating a computer inside your computer.
- I used virtualization mainly to test and install NixOS without messing up my actual OS. Since NixOS isn't like other Linux systems with a graphical installer, doing it in a VM gave me space to try, make mistakes, and learn. If something went wrong, I could just delete the VM and start fresh. Also, it's helpful because you can try things without needing to worry about damaging your hardware or system.
- I installed Oracle VirtualBox and created a VM with some base memory (RAM), CPUs, and a virtual hard disk. Then I booted up the NixOS ISO, partitioned the disk manually, formatted it as ext4, and installed NixOS through the usual steps. The whole thing happened inside the VM, so my real machine stayed untouched Ness. I also learned that NixOS can be a host itself; it can run other virtual machines on top of it by enabling things like libvirtd and using tools like virt-manager. You just add a few lines in the configuration.nix file and it sets it up for you. That is one more of the really neat parts of NixOS: once you figure it out, it's all based on one configuration file.

11) <u>Implementing ftruncate()</u>

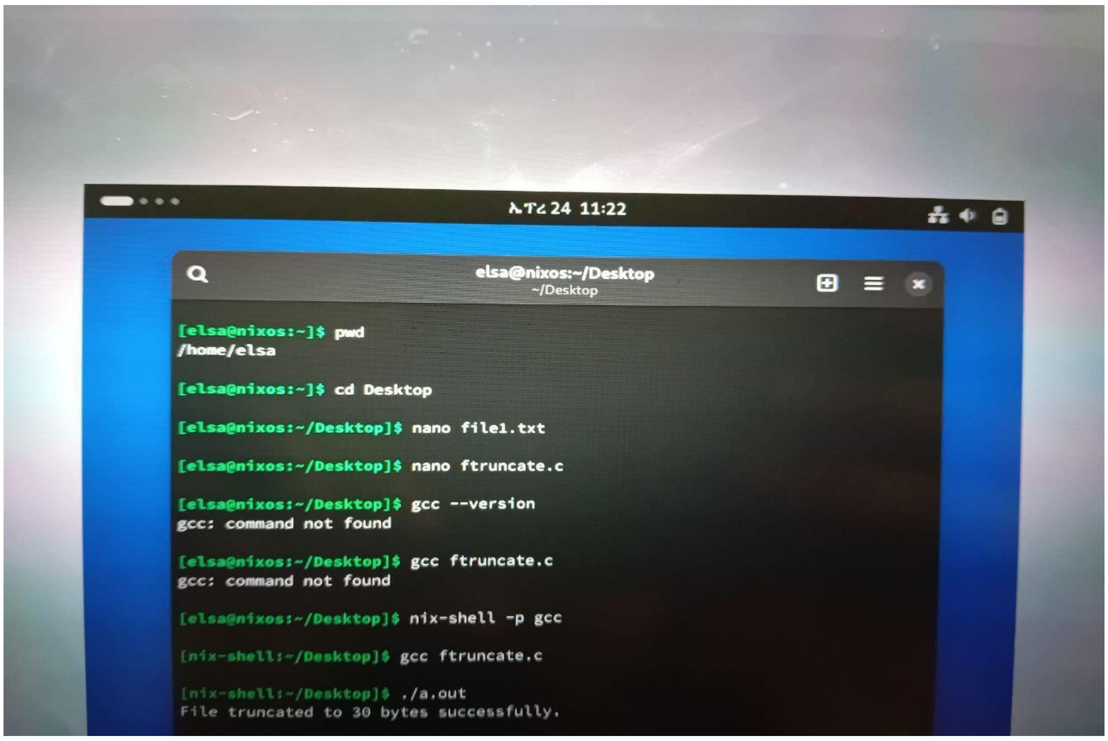
❖ In C, the ftruncate() function allows you to change the size of an open file. It comes in handy when you want to truncate or extend the file to a desired specific length. It belongs to the unistd.h library and works with file descriptors. When ftruncate(fd, size) is called, it instructs the system to truncate the file pointed to by the file descriptor fd to the new length size (in bytes). If the file was bigger, delete the rest; if smaller, the system would normally fill the extra space with null bytes (although this never happens in practice). Uses of ftruncate:

- To clean temporary files by strictly cutting the useful part.
- To reset or shrink logs, databases, or any kind of file.
- Useful in low-level system programming to get manual control over file size.

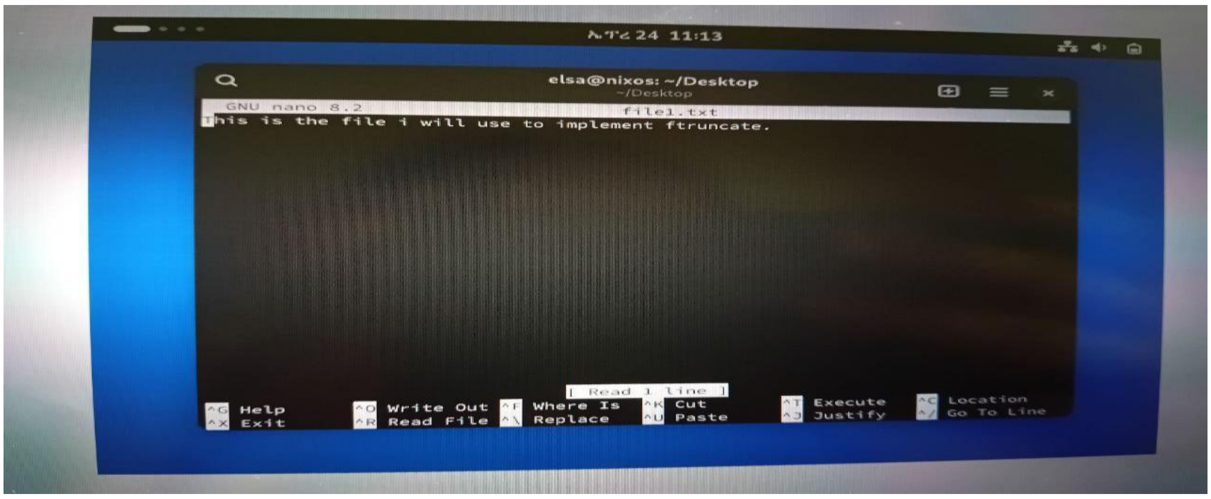
❖ Example :

```
int ftr=open("file1.txt",O_RDWR);

ftruncate(ftr,30);
```

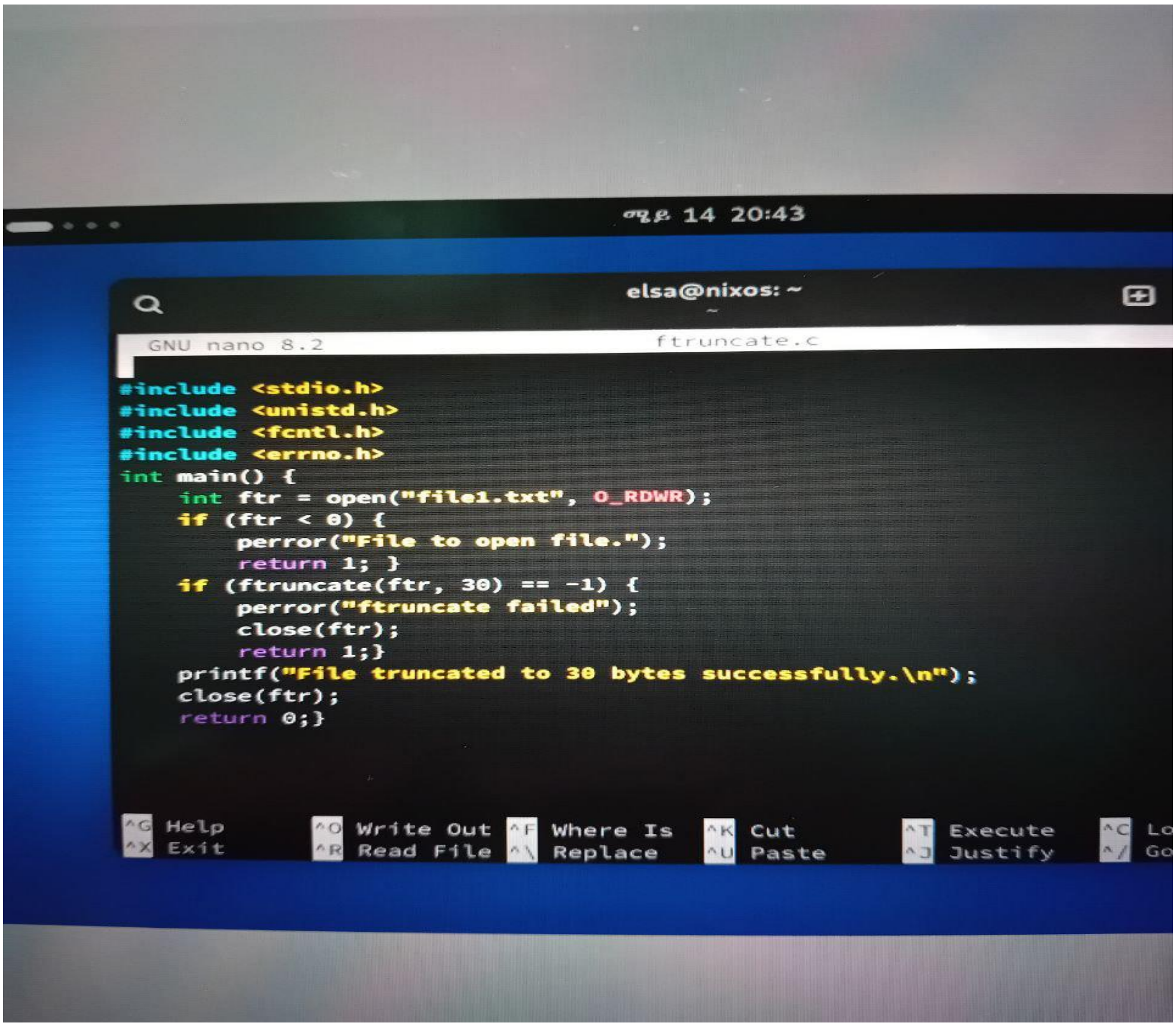
❖ Thus file "file1.txt" is opened and reshaped to 20 bytes:
❖ This means: "Keep the first 30 bytes of this file, and drop the rest."
❖ Here are the snipped images when I implement ftruncate() :



```
[elsa@nixos:~]$ pwd
/home/elsa

[elsa@nixos:~]$ cd Desktop

[elsa@nixos:~/Desktop]$ nano file1.txt

[elsa@nixos:~/Desktop]$ nano ftruncate.c

[elsa@nixos:~/Desktop]$ gcc --version
gcc: command not found

[elsa@nixos:~/Desktop]$ gcc ftruncate.c
gcc: command not found

[elsa@nixos:~/Desktop]$ nix-shell -p gcc

[nix-shell:~/Desktop]$ gcc ftruncate.c

[nix-shell:~/Desktop]$ ./a.out
File truncated to 30 bytes successfully.
```



```
ኤፕሪ 24 11:13
elsa@nixos: ~/Desktop
~/Desktop
  GNU nano 8.2                         file1.txt
This is the file i will use to implement ftruncate.

                              [ Read 1 line ]
^G Help      ^O Write Out  ^F Where Is  ^K Cut     ^T Execute  ^C Location
^X Exit      ^R Read File  ^\ Replace   ^U Paste   ^J Justify  ^/ Go To Line
```



```
ማርች 14 20:43
elsa@nixos: ~
~
  GNU nano 8.2                         ftruncate.c

#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
int main() {
    int ftr = open("file1.txt", O_RDWR);
    if (ftr < 0) {
        perror("File to open file.");
        return 1; }
    if (ftruncate(ftr, 30) == -1) {
        perror("ftruncate failed");
        close(ftr);
        return 1;}
    printf("File truncated to 30 bytes successfully.\n");
    close(ftr);
    return 0;}

^G Help      ^O Write Out  ^F Where Is  ^K Cut     ^T Execute  ^C Lo
^X Exit      ^R Read File  ^\ Replace   ^U Paste   ^J Justify  ^/ Go
```