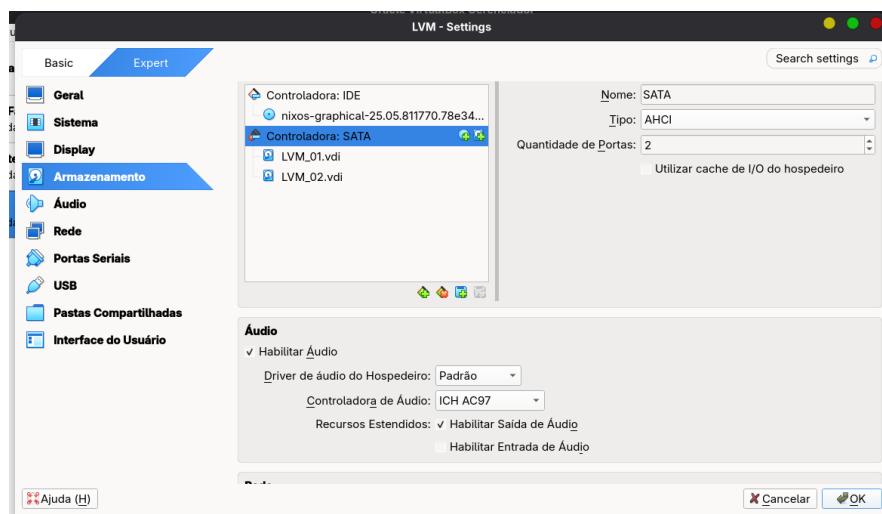


Instalando Nix OS manualmente.

Nix OS é uma distribuição Linux independente que possui pacotes próprios, uma linguagem de programação própria, o Nix, e um sistema de gerenciamento de pacotes declarativo, isto é, você instala os pacotes através de declarações num local específico do arquivo de configuração na linguagem nix, o /etc/nixos/configuration.nix. O Nix não é uma distribuição que se baseia na distribuição de arquivos no modelo padrão do Linux, e possui um diretório próprio, o nix, onde há uma pasta chamada store onde todos os pacotes são alocados. Por ser atômico, muitos podem pensar que não é possível configurar de forma adequada os discos as partições do sistema, mas pode sim e, isso é feito adequadamente na instalação manual. Os procedimentos a seguir foram feitos numa máquina virtual contendo dois discos de 50GB cada. O sistema de formatação de arquivos a ser utilizado será o padrão do Linux EXT4 que poderia ser substituído por um BTRFS ou qualquer outro que desejar. A formatação se dará em cima da camada do LVM, pois a dupla EXT4 e LVM, permite flexibilização de capacidade de armazenamento, aumentando ou diminuindo o tamanho das partições e ou, numa possibilidade diferente, adicionar mais discos sem grandes problemas, além de também poder tirar snapshots e realizar backups. Aliás, grande parte do trabalho de configuração fica em torno destas tarefas em específico. O Nix OS para esta tarefa é muito prático e acredito que isto faz com que seja bom para aqueles que gostam de configurar seu equipamento sem muitos passos.



Conforme imagem acima, a ISO usada foi a de modo gráfica, pois ela ajuda a navegar na rede, se for preciso e manipular certos arquivos mais eficiente.

A screenshot of a terminal window titled 'root@nixos: ~'. The command 'fdisk -l' was run, displaying information about three disks: /dev/loopb, /dev/sda, and /dev/sdb. The output is as follows:

```
[root@nixos:~]# fdisk -l
Disk /dev/loopb: 3.77 GiB, 4050223104 bytes, 7910592 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 44426862-585B-4888-8632-FB83E4582FCD

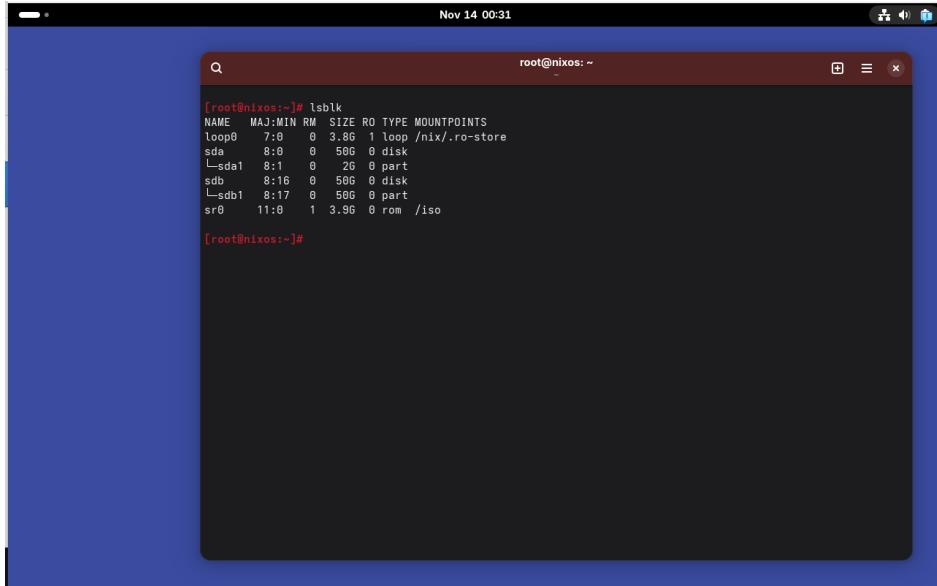
Device      Start    End Sectors Size Type
/dev/sda1   2048 4196351 4194304  26 EFI System

Disk /dev/sdb: 50 GiB, 53687091200 bytes, 104857600 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 0EE6944A-0AC3-4558-A6EB-4133A52B00A8

Device      Start    End Sectors Size Type
/dev/sdb1   2048 104855551 104853504  506 Linux LVM

[root@nixos:~]#
```

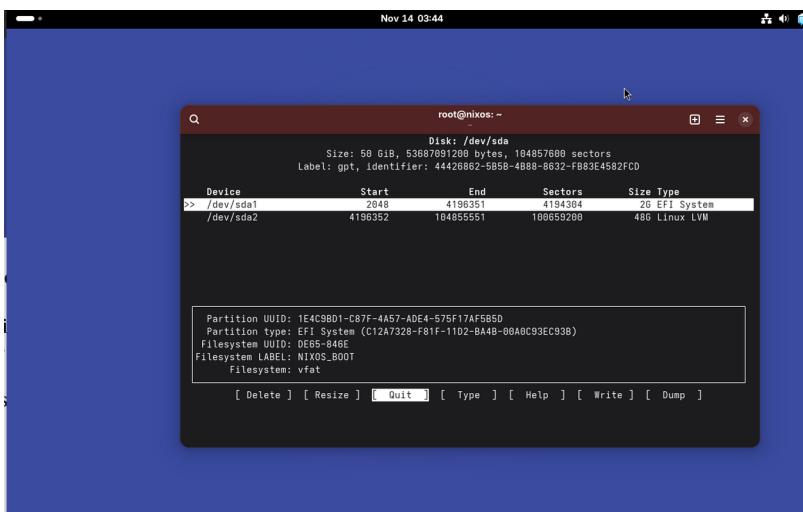
Conforme a segunda imagem, foi aberto o Terminal, (no KDE é o Konsole) e o controle foi passado para o administrador root através do comando sudo -i aplicado diretamente no terminal.



```
[root@nixos:~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0    7:0     0  3.8G  1 loop /nix/.ro-store
sda     8:0     0 500G  0 disk 
└─sda1   8:1     0   20G  0 part /
  └─sdb1   8:16   0  500G  0 disk 
    └─sdb1  8:17   0   500G  0 part /
sr0    11:0     1  3.9G  0 rom  /iso
```

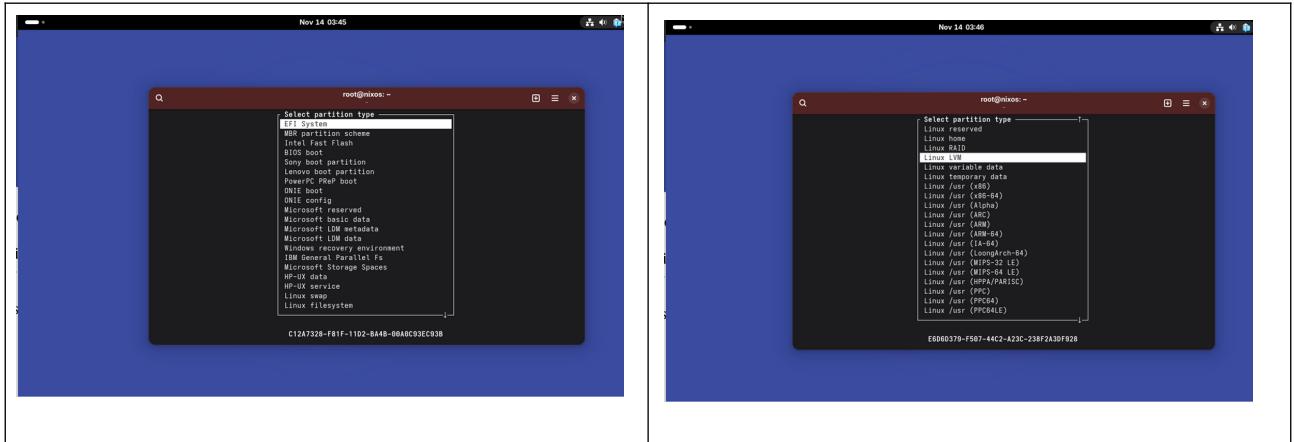
No exemplo, através do comando cfdisk, ou outro como o próprio fdisk, podemos criar as seguintes partições:

- No primeiro disco sda: a partição de inicialização do sistema que vai ser do tipo EFI de tamanho 2GB; a partição restante (48GB) que será utilizada como parte do grupo de volume geral;
- No segundo disco, apenas o disco inteiro como parte do volume geral.

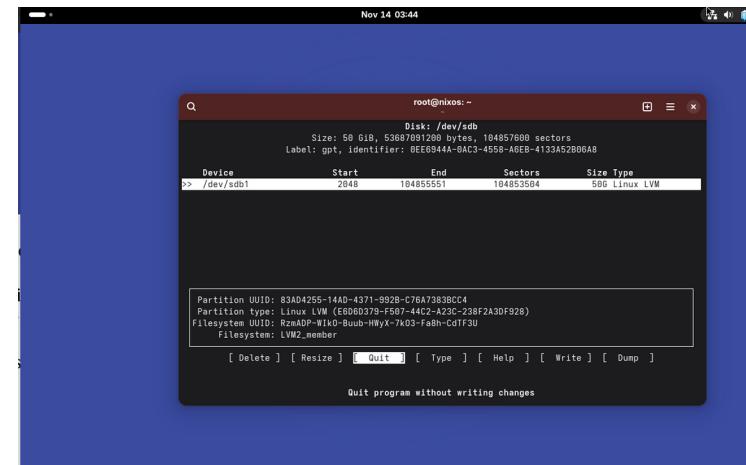


Ao iniciar o cfdisk, com o comando <cfdisk /dev/sda>, o mesmo pergunta se a tabela de partição a ser montada é a GPT, bastando apenas clicar na seleção. Acima o esquema do primeiro disco já montado, com o sistema de inicialização e tamanho de partição já definidos.

As figuras abaixo mostram as seleções para as duas partições.



Para o segundo disco, o esquema de partição ficará assim:



## LVM

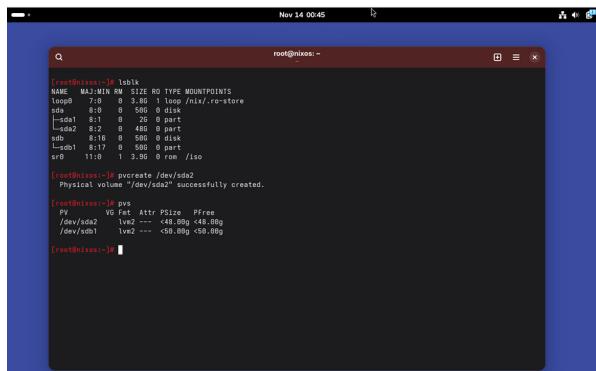
Passando esta etapa, os discos estão prontos para a configuração do LVM.

Para o particionamento com o LVM, há três etapas a serem seguidas:

1. A criação dos volumes físicos (/dev/sda2 e dev/sdb1);
2. A criação do grupo de volumes (nixlvm – único);
3. A criação dos volumes lógicos (root, swap, home).

Para a criação dos volumes físicos basta digitar o seguinte comando:

```
pvcreate /dev/sda2 /dev/sdb1
```



```
[root@nixos:~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0   7:0:0    0  1.8G  1 loop /nix/store
sda0   8:0:0    0 590G  0 disk
└─sda1  8:1:0    0  26G  0 part
  └─sda2  8:2:0    0  24G  0 part
  └─sda3  8:3:0    0  53G  0 part
sdb0   8:16:0   0 590G  0 disk
└─sdb1  8:17:0   0 590G  0 part
sr0   11:0:0    1  3.5G  0 rom /iso

[root@nixos:~]# pvcreate /dev/sda2
Physical volume "/dev/sda2" successfully created.
[root@nixos:~]# pvs
PV      VG   Attr PSize  PFree
/dev/sda2  nixv2 --- <48.80g <48.80g
/dev/sdb1  nixv2 --- <59.00g <59.00g
[root@nixos:~]
```

Para verificar a entrada dos dados basta digitar:  
pvs ou pvdisplay

Para a criação do Grupo de Volume, será necessário digitar:  
vgcreate nixlvm /dev/sda2 /dev/sdb1

Da mesma forma que o comando pvs e pvdisplay, o comando vgs e vgdisplay se encrregam da tarefa de mostrar os detalhes deste Grupo de Volume o nixlvm. Abaixo, a imagem dos detalhes deste comando e os próximos.

```
Nov 14 00:53
root@nixos: ~

--- Volume group ---
VG Name           nixlvm
System ID
Format            lvm2
Metadata Areas   2
Metadata Sequence No 1
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size          97.99 GiB
PE Size           4.00 MiB
Total PE          25086
Alloc PE / Size  0 / 0
Free  PE / Size  25086 / 97.99 GiB
VG UUID          yuJM8M-JMft-yHAL-YtWz-5Ye1-IHjM-Zx8Gpv

[root@nixos:~]# lvcreate -l 512 nixlvm -n swap
Logical volume "swap" created.

[root@nixos:~]# lvcreate -l 7680 nixlvm -n root
Logical volume "root" created.

[root@nixos:~]# lvcreate -l 768 nixlvm -n home
Logical volume "home" created.

[root@nixos:~]#
```

De acordo com os detalhes desta imagem podemos notar o valor total do grupo (97,99 GiB) e também o mesmo cálculo considerando o volume mínimo estendido de 4MiB, mostrando o valor 25086 PE. Este valor ou valores abaixo dele serão importantes para a criação dos volumes lógicos. No exemplo acima, foi utilizado o valor em torno de PEs para criar as partições. Note que o comando <lvcreate -l 512 nixlvm -n swap> é igual ao comando <lvcreate -L 2G nixlvm -n swap>, onde:

- -l para múltiplos de PEs;
- -L para valores diretos GiB ou similares;
- -n para os nomes dos volumes lógicos.

Assim como os comandos anteriores, o comando lvs e lvdisplay são equivalentes.

```
Nov 14 00:54
root@nixos: ~

[root@nixos:~]# pvs
PV   VG   Fmt Attr PSize  PFree
/dev/sda2  nixlvm lvm2 e-- <48.00g <3.00g
/dev/sdb1  nixlvm lvm2 e-- <50.00g >50.00g

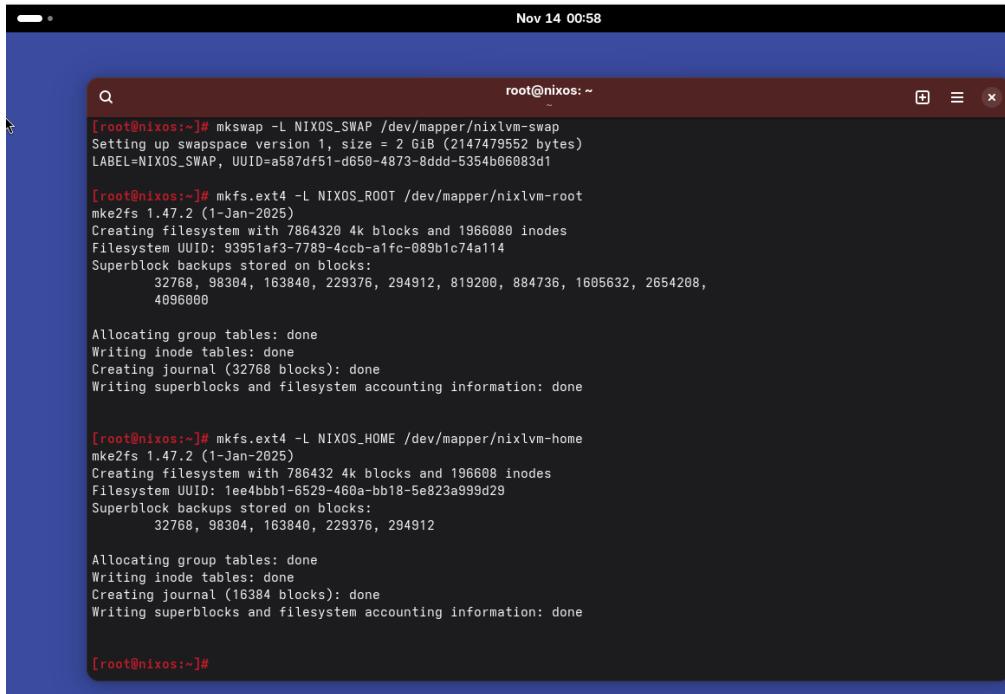
[root@nixos:~]# vgs
VG   #PV #LV #SN Attr  VSize  VFree
nixlvm  2   3   0  w--n-  97.99g 62.99g

[root@nixos:~]# lvs
LV   VG   Attr    LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
home nixlvm -wi-a---- 3.00g
root nixlvm -wi-a---- 50.00g
swap nixlvm -wi-a---- 2.00g

[root@nixos:~]#
```

## Tipo de Sistema de Arquivos

Após a criação dos volumes lógicos, será necessário criar o sistema de arquivos FAT32, EXT4 e SWAP para a montagem do sistema, conforme mostra a imagem abaixo.



The screenshot shows a terminal window titled "root@nixos: ~" running on a Linux system. The terminal displays the following commands and their outputs:

```
[root@nixos:~]# mkswap -L NIXOS_SWAP /dev/mapper/nixlvm-swap
Setting up swapspace version 1, size = 2 GiB (2147479552 bytes)
LABEL=NIXOS_SWAP, UUID=a587df51-d650-4873-8ddd-5354b06083d1

[root@nixos:~]# mkfs.ext4 -L NIXOS_ROOT /dev/mapper/nixlvm-root
mke2fs 1.47.2 (1-Jan-2025)
Creating filesystem with 7864320 4k blocks and 1966080 inodes
Filesystem UUID: 93951af3-7789-4ccb-a1fc-089b1c74a114
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[root@nixos:~]# mkfs.ext4 -L NIXOS_HOME /dev/mapper/nixlvm-home
mke2fs 1.47.2 (1-Jan-2025)
Creating filesystem with 786432 4k blocks and 196608 inodes
Filesystem UUID: 1ee4bbb1-6529-460a-bb18-5e823a999d29
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

[root@nixos:~]#
```

Os comandos são:

- mkfs.ext4 -L NIXOS\_ROOT /dev/mapper/nixlvm-root;
- mkfs.ext4 -L NIXOS\_HOME /dev/mapper/nixlvm-home;
- mkswap -L NIXOS\_SWAP /dev/mapper/nixlvm-swap;
- mkfs.fat -n NIXOS\_BOOT -F32 /dev/sda1.

## Montagem dos sistemas de partções

Com os sistemas de arquivos definidos, agora basta finalizar a montagem dos mesmos para seguir com a instalação em si. Os comando utilizados são:

- mkdir -p /mnt/{boot,home};
- mount /dev/mapper/nixlvm-root /mnt
- mount /dev/mapper/nixlvm-home /mnt/home
- mount -o umask=0077 /dev/sda1 /mnt/boot
- swapon /dev/mapper/nixlvm-swap

A seguir, imagem com os detalhes da montagem

```
[root@nixos:~]# mkfs.ext4 -L NIXOS_HOME /dev/mapper/nixlvm-home
mke2fs 1.47.2 (1-Jan-2025)
Creating filesystem with 786432 4k blocks and 196608 inodes
Filesystem UUID: 1ee4bbb1-6529-460a-bb18-5e023a999d29
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

[root@nixos:~]# mkdir -p /mnt/{boot,home}
[root@nixos:~]# mount /dev/mapper/nixlvm-root /mnt
[root@nixos:~]# mount /dev/mapper/nixlvm-home /home

[root@nixos:~]# mount -o umask=0077 /dev/sda1 /mnt/boot
mount: /mnt/boot: mount point does not exist.
        dmesg(1) may have more information after failed mount system call.

[root@nixos:~]# mkdir -p /mnt/boot
[root@nixos:~]# mount -o umask=0077 /dev/sda1 /mnt/boot
[root@nixos:~]# swapon /dev/mapper/nixlvm-swap
[root@nixos:~]#
```

Na imagem abaixo, atento a instrução para a montagem do diretório home. O comando correto é esse:

- `mount /dev/mapper/nixlvm-home /mnt/home`

Caso haja algum problema, basta corrigir digitação e realizar a geração da configuração do nix novamente `<nixos-generate-config -- root /mnt>`, como foi o meu caso.

```
[root@nixos:~]# mkdir -p /mnt/{boot,home}
[root@nixos:~]# mount /dev/mapper/nixlvm-root /mnt
[root@nixos:~]# mount /dev/mapper/nixlvm-home /home

[root@nixos:~]# mount -o umask=0077 /dev/sda1 /mnt/boot
mount: /mnt/boot: mount point does not exist.
        dmesg(1) may have more information after failed mount system call.

[root@nixos:~]# mkdir -p /mnt/boot
[root@nixos:~]# mount -o umask=0077 /dev/sda1 /mnt/boot
[root@nixos:~]# swapon /dev/mapper/nixlvm-swap

[root@nixos:~]# lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0  3.8G  1 loop /nix/.ro-store
sda       8:0    0  50G  0 disk 
└─sda1     8:1    0   2G  0 part /mnt/boot
  └─sda2     8:2    0  48G  0 part 
    ├─nixlvm-swap 254:0  0   2G  0 lvm  [SWAP]
    ├─nixlvm-root 254:1  0  30G  0 lvm  /mnt
    └─nixlvm-home 254:2  0   3G  0 lvm  /home
sdb       8:16   0  50G  0 disk 
└─sdb1     8:17   0  50G  0 part 
sr0      11:0    1  3.9G  0 rom /iso

[root@nixos:~]#
```

```

Nov 14 01:06
root@nixos: ~

[root@nixos:~]# mount /dev/mapper/nixlvm-home /home
[root@nixos:~]# mount -o umask=0077 /dev/sda1 /mnt/boot
mount: /mnt/boot: mount point does not exist.
      dmesg(1) may have more information after failed mount system call.

[root@nixos:~]# mkdir -p /mnt/boot
[root@nixos:~]# mount -o umask=0077 /dev/sda1 /mnt/boot
[root@nixos:~]# swapon /dev/mapper/nixlvm-swap
[root@nixos:~]# lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0  3.8G  1 loop /nix/.ro-store
sda       8:0    0  50G  0 disk
└─sda1     8:1    0   2G  0 part /mnt/boot
  └─sda2     8:2    0  48G  0 part
    ├─nixlvm-swap 254:0  0   2G  0 lvm  [SWAP]
    ├─nixlvm-root 254:1  0   30G  0 lvm  /mnt
    └─nixlvm-home 254:2  0   3G  0 lvm  /home
sdb       8:16   0  50G  0 disk
└─sdb1     8:17   0  50G  0 part
sr0      11:0    1  3.9G  0 rom  /iso

[root@nixos:~]# 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.

[root@nixos:~]#

```

Com o comando `nixos-generate-config --root /mnt`, há a geração dos arquivos `/mnt/etc/nixos/hardware-configuration.nix` e `/mnt/etc/nixos/configuration.nix`, e o sistema nix está com quase tudo pronto para ser instalado, faltando apenas detalhar o que será instalado e a suas configurações pessoais.

## Configurando os arquivos do Nix

No arquivo `hardware-configuration.nix`, confira o esquema criado com o comando `lsblk`. Foi assim que corrigi o meu problema anterior de montagem do `/mnt/home`. Desmontei o `/home` criado (`umount /dev/mapper/nixlvm-home`) para montá-lo no lugar certo (`/mnt/home`).

Dois pontos a considerar no arquivo `configuration.nix`:

- detalhes do tamanho de swap também é colocado nele;
- lembre-se de colocar um HASH da senha para o usuário logar inicialmente.

Para o segundo item acima, abra outro terminal (estando um já ocupado com a configuração) e digite:

- `sudo -i` (tecle enter)
- `# nix-shell --run 'mkpasswd -m SHA-512 -s' -p mkpasswd` (tecle enter)
- coloque a seguinte linha no arquivo `configuration.nix`:

`users.users.SEU_NOME.initialHashedPassword = "O_CÓDIGO_GERADO_NO COMANDO_ANTERIOR";`

- Crie a senha temporária quando no prompt aparecer `password`:
- Copie a senha em hash para o comando passado no arquivo `configuration.nix`.

Para o arquivo de trocas swap, basta acrescentar o seguinte no mesmo arquivo `configuration.nix`:

```
# Creating Swap Space
swapDevices = [{
```

```

device = "/swapfile";
size = 8 * 1024; # 8GB
}];
```

Segue abaixo os arquivos configuration.nix e hardware-configuration.nix na íntegra.

hardware-configuration.nix

```

# Do not modify this file! It was generated by 'nixos-generate-config'
# and may be overwritten by future invocations. Please make changes
# to /etc/nixos/configuration.nix instead.
{ config, lib, pkgs, modulesPath, ... }:

{
  imports = [ ];

  boot.initrd.availableKernelModules = [ "ata_piix" "ohci_pci" "ehci_pci" "ahci" "sd_mod"
"sr_mod" ];
  boot.initrd.kernelModules = [ "dm-snapshot" ];
  boot.kernelModules = [ "kvm-intel" ];
  boot.extraModulePackages = [ ];

  fileSystems."/" =
  { device = "/dev/disk/by-uuid/93951af3-7789-4ccb-a1fc-089b1c74a114";
    fsType = "ext4";
  };

  fileSystems."/boot" =
  { device = "/dev/disk/by-uuid/DE65-846E";
    fsType = "vfat";
    options = [ "fmask=0077" "dmask=0077" ];
  };

  fileSystems."/home" =
  { device = "/dev/disk/by-uuid/1ee4bbb1-6529-460a-bb18-5e823a999d29";
    fsType = "ext4";
  };

  swapDevices =
  [ { device = "/dev/disk/by-uuid/a587df51-d650-4873-8ddd-5354b06083d1"; }
];

# Enables DHCP on each ethernet and wireless interface. In case of scripted networking
# (the default) this is the recommended approach. When using systemd-networkd it's
# still possible to use this option, but it's recommended to use it in conjunction
# with explicit per-interface declarations with `networking.interfaces.<interface>.useDHCP`.
networking.useDHCP = lib.mkDefault true;
# networking.interfaces.enp0s3.useDHCP = lib.mkDefault true;

nixpkgs.hostPlatform = lib.mkDefault "x86_64-linux";
virtualisation.virtualbox.guest.enable = true;
}
```

## Configuration.nix

```
# Edit this configuration file to define what should be installed on
# your system. Help is available in the configuration.nix(5) man page, on
# https://search.nixos.org/options and in the NixOS manual (`nixos-help`).

{ config, lib, pkgs, ... }:

{
  imports =
    [ # Include the results of the hardware scan.
      ./hardware-configuration.nix
    ];
}

# Use the systemd-boot EFI boot loader.
boot.loader.systemd-boot.enable = true;
boot.loader.efi.canTouchEfiVariables = true;

networking.hostName = "nixos"; # Define your hostname.
# Pick only one of the below networking options.
# networking.wireless.enable = true; # Enables wireless support via wpa_supplicant.
networking.networkmanager.enable = true; # Easiest to use and most distros use this by default.

# Set your time zone.
time.timeZone = "America/Sao_Paulo";

# Configure network proxy if necessary
# networking.proxy.default = "http://user:password@proxy:port/";
# networking.proxy.noProxy = "127.0.0.1,localhost,internal.domain";

# Select internationalisation properties.
# i18n.defaultLocale = "en_US.UTF-8";
# console = {
#   font = "Lat2-Terminus16";
#   keyMap = "us";
#   useXkbConfig = true; # use xkb.options in tty.
# };

i18n.defaultLocale = "pt_BR.UTF-8";

i18n.extraLocaleSettings = {
  LC_ADDRESS = "pt_BR.UTF-8";
  LC_IDENTIFICATION = "pt_BR.UTF-8";
  LC_MEASUREMENT = "pt_BR.UTF-8";
  LC_MONETARY = "pt_BR.UTF-8";
  LC_NAME = "pt_BR.UTF-8";
  LC_NUMERIC = "pt_BR.UTF-8";
  LC_PAPER = "pt_BR.UTF-8";
  LC_TELEPHONE = "pt_BR.UTF-8";
  LC_TIME = "pt_BR.UTF-8";
}
```

```
};

# Enable the X11 windowing system.
services.xserver.enable = true;

# Enable the GNOME Desktop Environment.
services.xserver.displayManager.gdm.enable = true;
services.xserver.desktopManager.gnome.enable = true;

# Configure keymap in X11
services.xserver.xkb.layout = "br";
console.keyMap = "br-abnt2";
# services.xserver.xkb.options = "eurosign:e,caps:escape";

# Enable CUPS to print documents.
# services.printing.enable = true;

# Enable sound.
# services.pulseaudio.enable = true;
# OR
services.pipewire = {
    enable = true;
    alsa.enable = true;
    alsa.support32Bit = true;
    pulse.enable = true;
};

# Enable touchpad support (enabled default in most desktopManager).
# services.libinput.enable = true;

# Define a user account. Don't forget to set a password with 'passwd'.
users.users.SEUNOME_AQUI = {
    isNormalUser = true;
    extraGroups = [ "wheel" "libvirtd"]; # Enable 'sudo' for the user.
    packages = with pkgs; [
        # tree
    ];
    initialHashedPassword = "SUA_HASHED_SENHA_AQUI";
};

nixpkgs.config.allowUnfree = true;
# Installing Firefox
programs.firefox.enable = true;

# Installing Virt Manager
virtualisation.libvirtd = {
    enable = true;
```

```

qemu.vhostUserPackages = with pkgs; [ virtiofsd ];
};

programs.virt-manager.enable = true;

# Install Flatpak
services.flatpak.enable = true;

systemd.services.flatpak-repo = {
  wantedBy = [ "multi-user.target" ];
  path = [ pkgs.flatpak ];
  script = "flatpa remote-add --if-not-exists flathub https://dl.flathub.org/repo/flathub.flatpakrepo";
};

# List packages installed in system profile.
# You can use https://search.nixos.org/ to find more packages (and options).
environment.systemPackages = with pkgs; [
  vim # Do not forget to add an editor to edit configuration.nix! The Nano editor is also installed by default.
  wget
  git
  curl
  nixos-generators
];
swapDevices = [{ device = "/swapfile"; size = 2 * 1024; # 2GB }];
};

# Some programs need SUID wrappers, can be configured further or are
# started in user sessions.
# programs.mtr.enable = true;
# programs.gnupg.agent = {
#   enable = true;
#   enableSSHSupport = true;
# };

# List services that you want to enable:

# Enable the OpenSSH daemon.
# services.openssh.enable = true;

# Open ports in the firewall.
# networking.firewall.allowedTCPPorts = [ ... ];
# networking.firewall.allowedUDPPorts = [ ... ];
# Or disable the firewall altogether.
# networking.firewall.enable = false;

# Copy the NixOS configuration file and link it from the resulting system
# (/run/current-system/configuration.nix). This is useful in case you
# accidentally delete configuration.nix.
# system.copySystemConfiguration = true;

```

```

# This option defines the first version of NixOS you have installed on this particular machine,
# and is used to maintain compatibility with application data (e.g. databases) created on older
NixOS versions.
#
# Most users should NEVER change this value after the initial install, for any reason,
# even if you've upgraded your system to a new NixOS release.
#
# This value does NOT affect the Nixpkgs version your packages and OS are pulled from,
# so changing it will NOT upgrade your system - see https://nixos.org/manual/nixos/stable/#sec-
upgrading for how
# to actually do that.
#
# This value being lower than the current NixOS release does NOT mean your system is
# out of date, out of support, or vulnerable.
#
# Do NOT change this value unless you have manually inspected all the changes it would make to
your configuration,
# and migrated your data accordingly.
#
#      # For more information, see `man configuration.nix` or
https://nixos.org/manual/nixos/stable/options#opt-system.stateVersion .
system.stateVersion = "25.05"; # Did you read the comment?

}

```

## Instalação do sistema

Para incluir pacotes e configurações específicas, procure:

- Manual Nix, no cd da instalação;
- na web, Nix Package Search (<https://search.nixos.org/packages?channel=25.05>);
- na web, NixOS Wiki, ([https://wiki.nixos.org/wiki/NixOS\\_Wiki](https://wiki.nixos.org/wiki/NixOS_Wiki)).

Com os arquivos de configuração já modificados, e com os pacotes necessários, podemos prosseguir com a instalação do sistema. Esse passo é possível com o seguinte comando:

- # nixos-install –no-root-passwd
- # reboot

Conforme dito no começo do arquivo, a parte mais trabalhosa da instalação manual do NixOS vem antes, com o processo de montagem e escolha das partições. Este trabalho com o LVM e EXT4 valerá bastante, pois daqui para frente, será necessário só aumentar, reduzir ou anexar volumes lógicos. Abaixo, segue exemplos de como aumentar e diminuir os volumes lógicos.

- lvextend -l +768 /dev/nixlvm-root
- resize2fs /dev/mapper/nixlvm-root
- umount /dev/mapper/nixlvm-root
- e2fsck -f /dev/mapper/nixlvm-root
- resize2fs -p /dev/mapper/nixlvm-root 4G ou lvreduce -l -1024 (-1G) /dev/mapper/nixlvm-
root
- lvreduce -l 4136 (4G) /dev/mapper/nixlvm-root (TOTAL de %G para 4G)
- resize2fs /dev/mapper/nixlvm-root
- e2fsck -f /dev/mapper/nixlvm-root
- mount /dev/mapper/nixlvm-root

The terminal window shows the final steps of the NixOS installation:

```
Nov 14 02:24
root@nixos: ~
building '/nix/store/qnzzqmv080jbfz7ysqzzfv0p3rkhz5f-unit-reload-systemd-vconsole-setup.service.drv'...
building '/nix/store/jc5f05amljsj63546vz161ymqqlnk56b-wireplumber-configs.drv'...
building '/nix/store/mmygbj9j43fmpa56b6h27hsf0yf0n6m-tmpfiles.d.drv'...
building '/nix/store/ym0zzb1vds7c3q63f5dwa8inx5mana9-unit-display-manager.service.drv'...
created 2 symlinks in user environment
building '/nix/store/yvc7k8n6776ds7xh1ahcyi12nsmedzd-X-Restart-Triggers-systemd-tmpfiles-resetup.drv'...
building '/nix/store/j#mza0wq674rym6pq18gj92x56cznl-unit-wireplumber.service.drv'...
building '/nix/store/xyv1ra9ydhxxbbhs80gkvpczwkbdkh-unit-systemd-tmpfiles-resetup.service.drv'...
building '/nix/store/z181mbbd7h6n39dy2i6hhphwn83pfs-user-units.drv'...
building '/nix/store/jvhbmvmx2lppr2qs2mm5kh1b2m16-system-units.drv'...
building '/nix/store/5dlqfw6yxil0kjz75xjd9vjrh6z2151ax-etc.drv'...
building '/nix/store/6x145y6wcblwvgk38mp6hgc8riycaxc-nixos-system-nixos-25.05.811770.78e34d1667d3.drv'...
/nix/store/78wm5z761zy5wp05c2867f3vza8c4wc1-nixos-system-nixos-25.05.811770.78e34d1667d3
installing the boot loader...
setting up /etc...
Created "/boot/EFI".
Created "/boot/EFI/systemd".
Created "/boot/EFI/BOOT".
Created "/boot/loader".
Created "/boot/loader/keys".
Created "/boot/loader/entries".
Created "/boot/EFI/Linux".
Copied "/nix/store/d84f8nm2na5cr53m4jk0qk2mj7lgr9fx-systemd-257.9/lib/systemd/boot/efi/systemd-bootx64.efi" to "/boot/EFI/systemd/systemd-bootx64.efi".
Copied "/nix/store/d84f8nm2na5cr53m4jk0qk2mj7lgr9fx-systemd-257.9/lib/systemd/boot/efi/systemd-bootx64.efi" to "/boot/EFI/BOOT/BOOTX64.EFI".
Random seed file /boot/loader/random-seed successfully written (32 bytes).
Successfully initialized system token in EFI variable with 32 bytes.
Created EFI boot entry "Linux Boot Manager".
installation finished!
[root@nixos:~]#
```

Com a instalação finalizada, reinicie o sistema (reboot).

The terminal window shows the output of the `cat /etc/os-release` command and the `lsblk` command:

```
14 de nov 03:54
claudio@nixos: ~
[claudio@nixos:~]$ cat /etc/os-release
ANSI_COLOR="0;38;2;126;186;228"
BUG_REPORT_URL="https://github.com/NixOS/nixpkgs/issues"
BUILD_ID="25.05.811770.78e34d1667d3"
CPE_NAME="cpe:0:nixos:nixos:25.05"
DEFAULT_HOSTNAME=nixos
DOCUMENTATION_URL="https://nixos.org/learn.html"
HOME_URL="https://nixos.org/"
ID=nixos
ID_LIKE=""
IMAGE_ID=""
IMAGE_VERSION=""
LOGO="nix-snowflake"
NAME=NixOS
PRETTY_NAME="NixOS 25.05 (Warbler)"
SUPPORT_END="2025-12-31"
SUPPORT_URL="https://nixos.org/community.html"
VARIANT=""
VARIANT_ID=""
VENDOR_NAME=NixOS
VENDOR_URL="https://nixos.org/"
VERSION="25.05 (Warbler)"
VERSION_CODENAME=warbler
VERSION_ID="25.05"

[claudio@nixos:~]$ lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda        8:0    0   50G  0 disk
└─sda1     8:1    0   26G  0 part /boot
  └─sda2     8:2    0   48G  0 part
    ├─nixlvm-swap 254:0  0   2G  0 lvm  [SWAP]
    ├─nixlvm-root 254:1  0   30G  0 lvm  /nix/store
    └─nixlvm-home 254:2  0   3G  0 lvm  /home
sdb        8:16   0   50G  0 disk
└─sdb1     8:17   0   50G  0 part
sr0       11:0   1  3,9G  0 rom  /run/media/claudio/nixos-graphical-25.05-x86_64
```

Instalação concluída com sucesso.

Acredito que digitar este tutorial demorou bem mais que instalar o sistema em si!

Bons estudos e até mais.

Cláudio