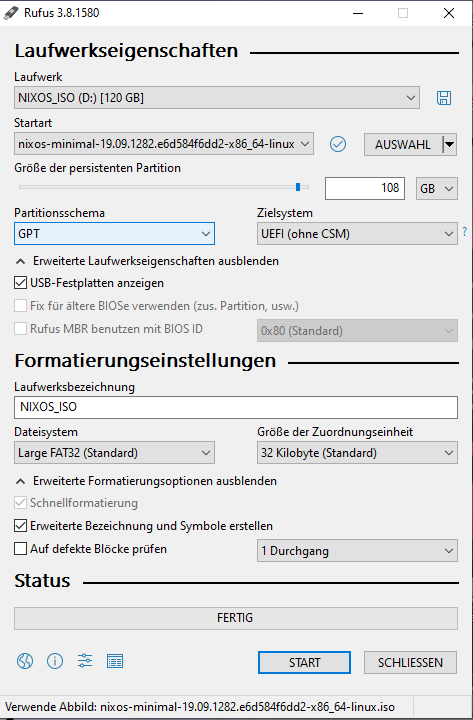
Installation of NixOS on external HDD

Firstly, I used the 64-bit minimal install CD. This is available on the [NixOS downloads page](https://nixos.org/nixos/download.html).

Once I’ve obtained the installation media, I copied it to my external HDD, which I connected with my notebook over a Sabrent cable. I used Rufus.exe to make a bootable HDD. **Note that this will destroy anything on the external HDD**.



Make sure, that the size of the bootable minimal nixos-partition is around 3 GB, the rest of the HDD is free. (Sorry, the picture of Rufus is in German).

After that, I opened the partition manager in Windows and checked, that the bootable NixOS-partition is around 3GB and the rest is free.

System configuration

Than I restart the computer. (I setup the BIOS boot-process that first it starts booting from external USB, second from internal HDD. So later, you just need to connect the external HDD and restart to start NixOS. If not connected the computer starts normal from your internal HDD to Windows. For me, that´s comfortable.)

There are some additional things in the UEFI system setup need to be configured for NixOS to install happily.

Once in the boot menu:

* Ensure safe boot is disabled
* Ensure UEFI mode is enabled

For the following steps I found a very helpful manual in the internet. I just followed these steps:

Installation

Now that your UEFI setup is configured, it’s time to boot the installation media.

Note that from here on in we’ll be in root prompts the whole time. The NixOS install environment helpfully drops you in a shell with root logged in.

Networking

Having internet access during an OS install can be handy to pull in configs. In the case of NixOS, if you want anything more than a very bare bones system to boot into, you’re going to want internet access to pull in system packages.

If you can’t just plug in an ethernet cable, then you’re probably going to want to use WiFi. To make that happen do the following:

-- Generates the actual key used to authenticate on your WPA secured network

# wpa\_passphrase $SSID $PASSPHRASE > /etc/wpa\_supplicant.conf

-- Restarts WPA Supplicant, which gives us WiFi for now

# systemctl restart wpa\_supplicant.service

(It is much easier, if you can connect your computer with a LAN-cable while install NixOS. I had a lot of problems configuring WLAN-access for the installation process.)

Partitioning

**This process will wipe anything on the disk**. Consider yourself warned.

As I understand it, a UEFI boot device requires a GUID partition table (GPT). Hence we’ll be using gdisk instead of the venerable fdisk. If you’re installing on a system that doesn’t use UEFI, then you can do a similar job with good ’ol fdisk.

To start, we’ll delete any existing partitions and start with a clean slate:

-- Identify the disk to install NixOS on - something like /dev/nvme0n1 or /dev/sda.

-- We'll refer to it as $DISK.

# lsblk

-- Open gdisk on the disk we're installing on

# gdisk $DISK

-----------------------

-- BEGIN GDISK COMMANDS

-- print the partitions on the disk

Command: p

-- Delete a partition. Select the partition number when prompted.

-- Repeat for all partitions.

Command: d

-- END GDISK COMMANDS

---------------------

**There, off course, you need to be very careful to choose the right diks and partition!!!**

In my case, I couldn´t delete ALL the partitions, because my partition 1 was my bootable NixOS-installation-file. We need to keep this!

Now we create the partitions we need: an EFI boot partition, and an LVM partition. LVM (logical volume management) allows us to more easily change our partitions (size and layout) should we need. In our case, the LVM partition will contain our root and swap partitions.

This code block assumes we’re still at a gdisk prompt.

-- Create the EFI boot partition

Command: n

Partition number: 1

First sector: <enter for default>

Last sector: +1G -- make a 1 gigabyte partition

Hex code or GUID: ef00 -- this is the EFI System type

-- Create the LVM partition

Command: n

Partition number: 2

First sector: <enter for default>

Last sector: <enter for default - rest of disk>

Hex code or GUID: 8e00 -- Linux LVM type

-- Write changes and quit

Command: w

**Note that because I kept my partition 1, the partition numbers for the rest of the manual is different. So add always +1 for the partition numbers (i.e. partition 2 in the manual is my partition 3)**

Encryption and LVM

Our partition table and primary partitions are in place. Now we can encrypt the partition that will contain our LVM partitions. This is the second partition that we created above - so should be something like /dev/nvme0n1p2 or /dev/sda2. We’ll refer to it as $LVM\_PARTITION below. Note that our boot partition won’t be encrypted. I can’t think of a reason why you would want this, and if you did, you probably wouldn’t need partitioning advice from me. Also note that our swap partition *is* encrypted. You don’t have any control over what’s moved into your swap space, so it could end up containing all sorts of private stuff in the clear - for example passwords copied from a password manager.

In our example below, we’re creating a swap space that is the same size as our RAM (16GB), and filling the rest of the disk with our root filesystem. You might want to tweak these sizes for your machine.

-- You will be asked to enter your passphrase - DO NOT FORGET THIS

# cryptsetup luksFormat $LVM\_PARTITION

-- Decrypt the encrypted partition and call it nixos-enc. The decrypted partition

-- will get mounted at /dev/mapper/nixos-enc

# cryptsetup luksOpen $LVM\_PARTITION nixos-enc

-- Create the LVM physical volume using nixos-enc

# pvcreate /dev/mapper/nixos-enc

-- Create a volume group that will contain our root and swap partitions

# vgcreate nixos-vg /dev/mapper/nixos-enc

-- Create a swap partition that is 16G in size - the amount of RAM on this machine

-- Volume is labeled "swap"'

# lvcreate -L 16G -n swap nixos-vg

-- Create a logical volume for our root filesystem from all remaining free space.

-- Volume is labeled "root"

# lvcreate -l 100%FREE -n root nixos-vg

**Note that my RAM was 12G instead 16G in the manual, so I setup 12G.**

Create our filesystems

In the below snippet, $BOOT\_PARTITION refers to the boot partition created above - something like /dev/sda1.

-- Create a FAT32 filesystem on our boot partition

# mkfs.vfat -n boot $BOOT\_PARTITION

-- Create an ext4 filesystem for our root partition

# mkfs.ext4 -L nixos /dev/nixos-vg/root

-- Tell our swap partition to be a swap

# mkswap -L swap /dev/nixos-vg/swap

-- Turn the swap on before install

# swapon /dev/nixos-vg/swap

Mount filesystems and prep for install

We’re almost there. Now it’s time to mount the partitions we’ve created, put our system configuration in place, and finally, pull the trigger.

The snippet below uses $BOOT\_PARTITION as a placeholder for the UEFI boot partition we created earlier. This was the first partition on the disk, and will probably be something like /dev/sda1 or /dev/nvme0n1p1.

# mount /dev/nixos-vg/root /mnt

# mkdir /mnt/boot

# mount $BOOT\_PARTITION /mnt/boot

Now that we have filesystems we can write to, let’s generate our initial config.

# nixos-generate-config --root /mnt

Configuration

NixOS is primarily configured by /etc/nixos/configuration.nix. Given that our root filesystem is mounted at /mnt, that will be /mnt/etc/nixos/configuration.nix for now. Let’s open it up and set some important options.

If anything is broken in your config, installation should fail with an error message to help diagnose your problem. Furthermore, because NixOS is the way it is, you can radically reconfigure your system later knowing that you can fallback to a known good configuration, and once you’re confident everything works, clean up packages you no longer need. In short, don’t stress too much about installing and configuring absolutely everything. It’s fine to start with a small but working system and build up as you learn what you want.

-- Vim 4 life! Or, you know, use `nano` or whatever else you might prefer.

vim /mnt/etc/nixos/configuration.nix

It is of critical importance that we tell NixOS we have a LUKS encrypted partition that needs to be decrypted before we can access any LVM partitions. We do that like so.

boot.initrd.luks.devices = [

{

name = "root";

device = "/dev/nvme0n1p2";

preLVM = true;

}

];

NixOS also needs to know that we’re using EFI, however this was correctly configured for me automatically.

boot.loader.systemd-boot.enable = true;

I also use network manager and its associated applet to manage my networking. If you’d like to do the same, add the following, as well as the applet package mentioned below.

networking.networkmanager.enable = true;

In addition to these core configuration items, you might want to install some packages to get you started. Our NixOS install will be very bare without them. Packages can be specified as additional configuration items, and there should be a commented out section of configuration that you can uncomment and edit. For example, a fairly modest set of packages would look something like this. Note that networkmanagerapplet is included to give us a tray icon to configure networking from.

As the comment in the configuration file tells you, you can search for packages to install with nix-env -qaP | grep $PACKAGE.

environment.systemPackages = (with pkgs; [

firefox

git

htop

networkmanagerapplet

nix-prefetch-scripts

nix-repl

vagrant

vim

wget

which

xscreensaver

]);

One last thing I’ll call out, is specifying your user. It’s not a good idea to use root all the time, so to create your user, add/uncomment something like the following. In the example below, we’ll create a user called “holochain”. We’ll give them a home directory and add them to a few groups. Most importantly, you probably want your user to be a member of wheel so they can run privileged commands with sudo.

users.extraUsers.holochain = {

createHome = true;

extraGroups = ["wheel" "video" "audio" "disk" "networkmanager"];

group = "users";

home = "/home/holochain";

isNormalUser = true;

uid = 1000;

};

By default you’ll get [Plasma](https://www.kde.org/plasma-desktop) as your desktop environment. If you want something else, then you’ll have to do some research on what’s available and how to configure it.

There’s a bunch of other stuff commented out in the generated configuration.nix and I encourage you to read through it and uncomment and/or set anything that takes your fancy. For example, setting your time zone is probably a good idea.

Configuration files may vary between NixOS versions, so be sure to check that there are no version specific subtleties before borrowing heavily from another configuration.nix file.

Pull the trigger!

Once you’re happy with your configuration, we can pull the trigger on an install.

# nixos-install

-- IT'LL ASK YOU FOR YOUR ROOT PASSWORD NOW - DON'T FORGET IT

# reboot

Go get a coffee while everything installs, and hopefully you’ll reboot to your new system.

If something has gone wrong, don’t worry. You can always boot back into the installation media, mount your partitions, update the configuration, and install again. To mount existing partitions, you’ll need to decrypt the LVM partition and then activate its volume group.

# cryptsetup luksOpen $LVM\_PARTITION nixos-enc

# lvscan

# vgchange -ay

# mount /dev/nixos-vg/root /mnt

# ...

Assuming your system has booted to a login screen, you’re going to want to set your user’s password so you don’t login to your graphical environment as root. To do this, press Ctrl-Alt-F1 to open a terminal, login as root, and run passwd $USER, replacing $USER with the name of the user you configured. Once set, run reboot to reboot your machine and login as your regular user.

**For me, this works great! After installed NixOS and its packages through a LAN-cable I started NixOS´s graphical environment with “systemctl start display-manager” and configured**

* **WLAN-access**
* **German keyboard**
* **time**