Install NixOS with disko disk partitioning

In this second tutorial, we will walk you through the process of installing NixOS. Unlike the first installation tutorial, we will use the command line to install NixOS manually, except for using disko to specify disk partitions declaratively in Nix itself. This is the first step toward our next tutorial, where we will automate the entire installation process.

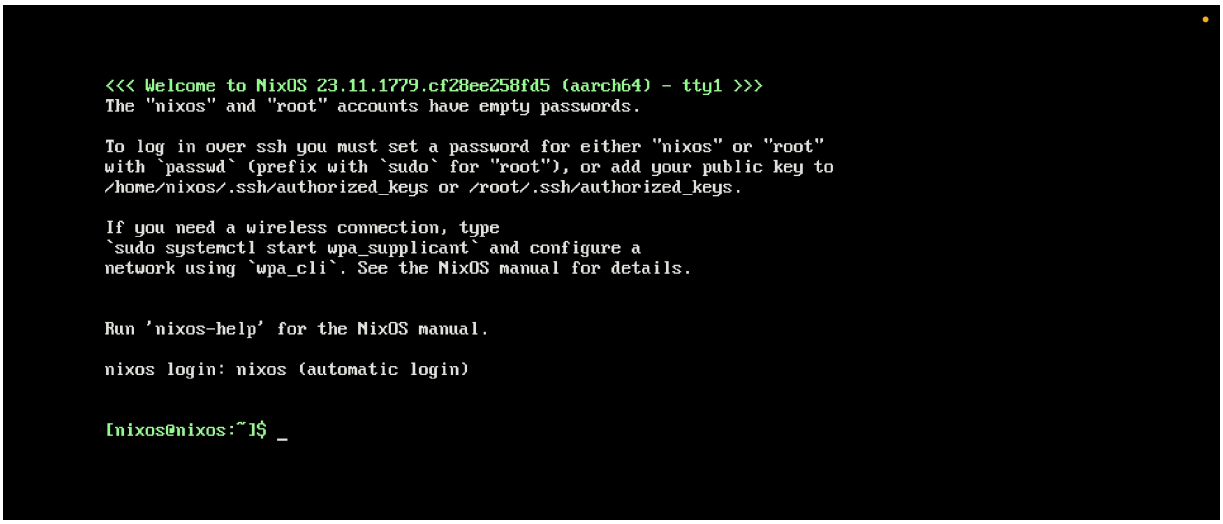
Prepare to install NixOS

Minimal ISO image

This tutorial doesn't use a graphical installer. Instead, it uses the minimal ISO image. This is primarily because we don't want the installer to partion the disk for us. We will use disko to do that.

- Download the latest NixOS ISO from here. Choose the "Minimal ISO image" for your architecture.
- Create a bootable USB flash drive (instructions here)

Boot your computer from this USB flash drive, and expect to be greeted with a command line interface (CLI):



Partition the disk

Before installing NixOS, let's define our partition layout in Nix. We will follow the official disko documentation and include screenshots wherever necessary. Finally, we will use flakes to manage the configuration.

Choosing the disk configuration

Instead of creating our partition layout from scratch, we can choose one of the examples Disko itself provides (see here). We will use the hybrid example as it will work for both BIOS and UEFI systems.

Retrieve the disk configuration to a temporary location, calling it `disko-config.nix` (we will use it latter):

```
curl https://raw.githubusercontent.com/nix-community/disko/master/example/hybr
```

Modify the disk configuration

The above downloaded Nix file uses a hardcoded disk device. So, we need to replace it with the device name of the disk we want to install NixOS on. We can use `lsblk` to find it.

```
[nixos@nixos:~]$ lsblk
NAME  MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0   7:0    0 871.7M  1 loop /nix/.ro-store
sr0    11:0    1 945.9M  0 rom  /iso
vda   254:0    0    64G  0 disk
```

In this case, the device name is `vda`. The device file is located at `/dev/vda`. We will use this to modify `disko-config.nix` we downloaded earlier.

```
{
  disko.devices = {
    disk = {
      main = {
        type = "disk";
        device = "/dev/vda";
        content = {
          type = "gpt";
          partitions = {
            boot = {
              size = "1M";
              type = "EF02"; # for grub MBR
            };
            ESP = {
              size = "512M";
              type = "EF00";
              content = {
                type = "filesystem";
                format = "vfat";
                mountpoint = "/boot";
              };
            };
            root = {
              size = "100%";
              content = {
                type = "filesystem";
                format = "ext4";
                mountpoint = "/";
              };
            };
          };
        };
      };
    };
  };
}
"disko-config.nix" 37L, 782B written
```

Run the partitioning script

The disko flake provides an app that will take our partitioning scheme defined in Nix file above, partition the specified disk device and mount it at `/mnt`. We want this to happen prior to installing NixOS. Let's do that now:

```
sudo nix \
  --experimental-features "nix-command flakes" \
  run github:nix-community/disko -- \
  --mode disko /tmp/disko-config.nix
```

Once the command completes, you should see the disk partitioned and mounted at `/mnt`:

```
[nix-shell:/tmp]$ mount | grep /mnt
overlay on /nix/store type overlay (rw,relatime,lowerdir=/mnt-root/nix/.ro-store,upperdir=/mnt-root/
nix/.rw-store/store,workdir=/mnt-root/nix/.rw-store/work)
overlay on /nix/store type overlay (ro,relatime,lowerdir=/mnt-root/nix/.ro-store,upperdir=/mnt-root/
nix/.rw-store/store,workdir=/mnt-root/nix/.rw-store/work)
/dev/vda3 on /mnt type ext4 (rw,relatime)
/dev/vda1 on /mnt/boot type vfat (rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=iso8859-1
,shortname=mixed,errors=remount-ro)

[nix-shell:/tmp]$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0 871.7M  1 loop /nix/.ro-store
sr0     11:0    1 945.9M  0 rom  /iso
vda    254:0    0    64G  0 disk
├vda1 254:1    0   512M  0 part /mnt/boot
├vda2 254:2    0     1M  0 part
└vda3 254:3    0  63.5G  0 part /mnt
```

Generate initial NixOS configuration

With the disk partitioned, we are ready to follow the usual NixOS installation process. The first step is to generate the initial NixOS configuration under `/mnt`.

```
sudo nixos-generate-config --no-filesystems --root /mnt
```

Why `--no-filesystems` and `--root`?

- The fileSystems configuration will automatically be added by `disko`'s nixosModule (see below). Therefore, we use `--no-filesystems` to avoid generating it here.
- `--root` is to specify the mountpoint to generate `configuration.nix` and `hardware-configuration.nix` in. Here, our configuration will be generated in `/mnt/etc/nixos`.

Flakeify the configuration

Before we can utilize `disko` in our generated configuration, we will convert our configuration to a flake. This is a simple process of adding a `flake.nix` file in `/mnt/etc/nixos`:

```
# /mnt/etc/nixos/flake.nix
{
  inputs = {
    # NOTE: Replace "nixos-23.11" with that which is in system.stateVersion o
    # configuration.nix. You can also use latter versions if you wish to
    # upgrade.
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-23.11";
  };
  outputs = inputs@{ self, nixpkgs, ... }: {
    # NOTE: 'nixos' is the default hostname set by the installer
    nixosConfigurations.nixos = nixpkgs.lib.nixosSystem {
      # NOTE: Change this to aarch64-linux if you are on ARM
      system = "x86_64-linux";
      modules = [ ./configuration.nix ];
    };
  };
}
```

Make sure to change a couple of things in the above snippet:

- Replace `nixos-23.11` with the version from `system.stateVersion` in your `/mnt/etc/nixos/configuration.nix`. If you wish to upgrade right away, you can also use latter versions, or use `nixos-unstable` for the bleeding edge.
- `x86_64-linux` should be `aarch64-linux` if you are on ARM

For details, see Convert `configuration.nix` to be a flake.

Add the `disko` nixosModule

Our NixOS configuration still does not know anything about filesystems. Let's teach it just that by using the previously downloaded disko example. We do this by adding the `disko` flake input, importing its NixOS module before importing our `/tmp/disko-config.nix`.

1. Add the `disko` flake input in `/mnt/etc/nixos/flake.nix`:

```
# In `/mnt/etc/nixos/flake.nix`
{
  inputs = {
    disko.url = "github:nix-community/disko";
    disko.inputs.nixpkgs.follows = "nixpkgs";
  };
}
```

Why the "follows"?

`disko.inputs.nixpkgs.follows = "nixpkgs";` is to ensure that `disko` uses the same version of `nixpkgs` as specified in the current flake. This avoids having two different sources of `nixpkgs` and saves space.

2. Add the `disko` nixosModule:

```
{
  # In `outputs` of `/mnt/etc/nixos/flake.nix`
  nixosConfigurations.nixos = {
    # ...
    modules = [
      ./configuration.nix
      inputs.disko.nixosModules.disko
    ];
```

```
  };
}
```

```
{
  inputs = {
    nixpkgs.url = "github:nixos/nixpkgs/nixpkgs-unstable";
    disko.url = "github:nix-community/disko";
    disko.inputs.nixpkgs.follows = "nixpkgs";
  };
  outputs = { self, nixpkgs, disko }: {
    nixosConfigurations.nixos = nixpkgs.lib.nixosSystem {
      system = "aarch64-linux";
      modules = [ ./configuration.nix disko.nixosModules.disko ];
    };
  };
}
~
~
~
```

3.

Move the `disko-config.nix` to the flake directory:

```
mv /tmp/disko-config.nix /mnt/etc/nixos
```

4.

Add the disk configuration and use GRUB:

```
{
  # In `/mnt/etc/nixos/configuration.nix`
  imports = [
    ./hardware-configuration.nix
    ./disko-config.nix
  ];
  #boot.loader.systemd-boot.enable = true;
  #boot.loader.efi.canTouchEfiVariables = true;
  boot.loader.grub.enable = true;
  boot.loader.grub.efiSupport = true;
  boot.loader.grub.efiInstallAsRemovable = true;
}
```

```
  # 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
        ./disko-config.nix
      ];

    # Use the systemd-boot EFI boot loader.
    # boot.loader.systemd-boot.enable = true;
    # boot.loader.efi.canTouchEfiVariables = true;
    boot.loader.grub.enable = true;
    boot.loader.grub.efiSupport = true;
    boot.loader.grub.efiInstallAsRemovable = 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 = "Europe/Amsterdam";

    # 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";
"configuration.nix" 121L, 4208B written
```

Info

The boot loader configuration above is compatible with both BIOS and UEFI systems. Additionally, BIOS also requires boot.loader.grub.device to be set which is done by `disko`'s `nixosModule`.

Let's check that our final configuration is correct by using nix repl. In particular, we test the `fileSystems` set by `disko`:

```
# First, create a flake.lock
sudo nix --experimental-features "nix-command flakes" flake lock

# Start repl
nix --experimental-features "nix-command flakes" repl
```

```
Welcome to Nix 2.18.1. Type :? for help.

nix-repl> :lf .
Added 9 variables.

nix-repl> outputs.nixosConfigurations.nixos.config.fileSystems
{ "/" = { ... }; "/boot" = { ... }; }

nix-repl> outputs.nixosConfigurations.nixos.config.fileSystems."/"
{ autoFormat = false; autoResize = false; depends = [ ... ]; device = "/dev/disk/by-partlabel/disk-m
ain-root"; encrypted = { ... }; formatOptions = null; fsType = "ext4"; label = null; mountPoint = "/
"; neededForBoot = false; noCheck = false; options = [ ... ]; stratis = { ... }; }

nix-repl> outputs.nixosConfigurations.nixos.config.fileSystems."/boot"
{ autoFormat = false; autoResize = false; depends = [ ... ]; device = "/dev/disk/by-partlabel/disk-m
ain-ESP"; encrypted = { ... }; formatOptions = null; fsType = "vfat"; label = null; mountPoint = "/b
oot"; neededForBoot = false; noCheck = false; options = [ ... ]; stratis = { ... }; }
```

If you see something similar to the above, everything's good and we are ready to perform the actual installation.

Install NixOS

With our NixOS configuration in place, we will use the `nixos-install` program to install NixOS:

```
sudo nixos-install --root /mnt --flake '/mnt/etc/nixos#nixos'
# NOTE: You will be prompted to set the root password at this point.
sudo reboot
```

Once rebooted, you should be greeted with the NixOS login screen, allowing you to login to the machine using the root password you had set.

Bonus steps

This tutorial focused mostly on disko, but left some of the things covered in the previous tutorial which you might want to consider:

- Move configuration to home dir
- Store the configuration on Git
- Enable flakes

Video walkthrough

Video demo of the install 🔁 pic.twitter.com/KJLntZ6CrY
— Sridhar Ratnakumar (@sridca) February 19, 2024

Recap & next steps

You now have a reproducible disk partitioning scheme. This does come at the cost of a few extra manual steps, but you can automate them with a script. Which is what we will do in the next tutorial. We will use nixos-anywhere to automate the steps above and eliminate the need for a USB flash drive (assuming you have a working Linux system or are booted into a rescue image).

Links to this page

NixOS Tutorial Series

> ☑ Install NixOS with **disko** disk partitioning

Install NixOS with Flake configuration on Git

> In the next tutorial, we will automate the install process a bit by declaratively specifying our disk partitioning in Nix.

Install NixOS directly from a remote flake

> Unlike the previous tutorials (1; 2), the goal here is to near-fully automate our NixOS install using one command (see the next section).