NixOS virtual machines

Contents

One of the most important features of NixOS is the ability to configure the entire system declaratively, including packages to be installed, services to be run, as well as other settings and options.

NixOS configurations can be used to test and use NixOS using a virtual machine, independent of an installation on a "bare metal" computer.

What will you learn?

This tutorial serves as an introduction creating NixOS virtual machines. Virtual machines are a practical tool for experimenting with or debugging NixOS configurations.

What do you need?

- A Linux system with virtualisation support
- (optional) A graphical environment for running a graphical virtual machine
- A working Nix installation
- Basic knowledge of the Nix language

Skip to main content

Important

A NixOS configuration is a Nix language function following the NixOS module convention. For a thorough treatment of the module system, check the Module system deep dive tutorial.

Starting from a default NixOS configuration

Note

You can also skip this section and copy the sample configuration for this tutorial into a file `configuration.nix` in the current directory.

Use the `nixos-generate-config` command to create a configuration file that contains some useful defaults and configuration suggestions. The configuration produced from the following setup also is used for the NixOS minimal ISO image:

```
nix-shell -I nixpkgs=channel:nixos-24.05 -p "$(cat <<EOF
  let
    pkgs = import <nixpkgs> { config = {}; overlays = []; };
    iso-config = pkgs.path + /nixos/modules/installer/cd-dvd/installation-cd-minima
    nixos = pkgs.nixos iso-config;
  in nixos.config.system.build.nixos-generate-config
EOF
)"
```

Detailed explanation

Create a NixOS configuration in your working directory:

```
[nix-shell:~]$ nixos-generate-config --dir ./
```

Note

By default, when no `--dir` is specified, the generated configuration file is written to `/etc/nixos/configuration.nix`, which typically requires `sudo` permissions.

In the working directory you will then find two files:

Skip to main content

1. `hardware-configuration.nix` is specific to the hardware `nix-generate-config` is being run on.

   You can ignore that file for this tutorial because it has no effect inside a virtual machine.

2. `configuration.nix` contains various suggestions and comments for the initial setup of a desktop computer.

The default NixOS configuration without comments is:

```
{ config, pkgs, ... }:
{
  imports =  [ ./hardware-configuration.nix ];

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

  system.stateVersion = "24.05";
}
```

To be able to log in, add the following lines to the returned attribute set:

```
  users.users.alice = {
    isNormalUser = true;
    extraGroups = [ "wheel" ];
  };
```

Note

A configuration generated with `nixos-generate-config` contains this user configuration commented out.

Additionally, you need to specify a password for this user. For the purpose of demonstration only, you specify an insecure, plain text password by adding the `initialPassword` option to the user configuration:

```
    initialPassword = "test";
```

We add two lightweight programs as an example:

```
  environment.systemPackages = with pkgs; [
    cowsay
    lolcat
```

Skip to main content

⚠ Warning

Do not use plain text passwords outside of this example unless you know what you are doing. See `initialHashedPassword` or `ssh.authorizedKeys` for more secure alternatives.

This tutorial focuses on testing NixOS configurations on a virtual machine. Therefore you will remove the reference to `hardware-configuration.nix`:

```
-  imports =  [ ./hardware-configuration.nix ];
```

Sample configuration

The complete `configuration.nix` file looks like this:

```
{ config, pkgs, ... }:
{
  boot.loader.systemd-boot.enable = true;
  boot.loader.efi.canTouchEfiVariables = true;

  users.users.alice = {
    isNormalUser = true;
    extraGroups = [ "wheel" ]; # Enable ‘sudo’ for the user.
    initialPassword = "test";
  };

  environment.systemPackages = with pkgs; [
    cowsay
    lolcat
  ];

  system.stateVersion = "24.05";
}
```

Creating a QEMU based virtual machine from a NixOS configuration

A NixOS virtual machine is created with the `nix-build` command:

```
$ nix-build '<nixpkgs/nixos>' -A vm -I nixpkgs=channel:nixos-24.05 -I nixos-config=
```

This command builds the attribute `vm` from the `nixos-24.05` release of NixOS, using the NixOS configuration as specified in the relative path.

Detailed explanation

The complete `configuration.nix` file looks like this:

```
{ config, pkgs, ... }:
{
  boot.loader.systemd-boot.enable = true;
  boot.loader.efi.canTouchEfiVariables = true;

  services.xserver.enable = true;

  services.xserver.displayManager.gdm.enable = true;
  services.xserver.desktopManager.gnome.enable = true;

  users.users.alice = {
    isNormalUser = true;
    extraGroups = [ "wheel" ];
    initialPassword = "test";
  };

  system.stateVersion = "24.05";
}
```

To get graphical output, run the virtual machine without special options:

```
$ nix-build '<nixpkgs/nixos>' -A vm -I nixpkgs=channel:nixos-24.05 -I nixos-config=
$ ./result/bin/run-nixos-vm
```

Running Sway as Wayland compositor on a VM

To change to a Wayland compositor, disable `services.xserver.desktopManager.gnome` and enable `programs.sway`:

configuration.nix

```
-   services.xserver.desktopManager.gnome.enable = true;
+   programs.sway.enable = true;
```

Skip to main content

Note

Running Wayland compositors in a virtual machine might lead to complications with the display drivers used by QEMU. You need to choose from the available drivers one that is compatible with Sway. See QEMU User Documentation for options. One possibility is the `virtio-vga` driver:

```
$ ./result/bin/run-nixos-vm -device virtio-vga
```

Arguments to QEMU can also be added to the configuration file:

```
{ config, pkgs, ... }:
{
  boot.loader.systemd-boot.enable = true;
  boot.loader.efi.canTouchEfiVariables = true;

  services.xserver.enable = true;

  services.xserver.displayManager.gdm.enable = true;
  programs.sway.enable = true;

  imports = [ <nixpkgs/nixos/modules/virtualisation/qemu-vm.nix> ];
  virtualisation.qemu.options = [
    "-device virtio-vga"
  ];

  users.users.alice = {
    isNormalUser = true;
    extraGroups = [ "wheel" ];
    initialPassword = "test";
  };

  system.stateVersion = "24.05";
}
```

The NixOS manual has chapters on X11 and Wayland listing alternative window managers.

References

- NixOS Manual: NixOS Configuration.
- NixOS Manual: Modules.
- NixOS Manual Options reference.
- NixOS Manual: Changing the configuration.

Skip to main content

- NixOS source code: `vm` attribute in `default.nix`.
- Nix manual: `nix-build`.
- Nix manual: common command-line options.
- QEMU User Documentation for more runtime options
- NixOS option search: `virtualisation.qemu` for declarative virtual machine configuration

Next steps

- Module system deep dive
- Integration testing with NixOS virtual machines
- Building a bootable ISO image