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 working Nix installation on Linux, or NixOS, with a graphical environment
- Basic knowledge of the Nix

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

In this tutorial you will use a default configuration that is shipped with NixOS. You can also skip this section and copy the sample configuration for this tutorial into a file `configuration.nix` in the current directory.

Skip to main content

NixOS

On NixOS, use the `nixos-generate-config` command to create a configuration file that contains some useful defaults and configuration suggestions.

Beware that the result of this command depends on your current NixOS configuration. The output of `nixos-generate-config` can be made reproducible in a `nix-shell` environment. Here we provide a configuration that is used for the NixOS minimal ISO image:

```
nix-shell -I nixpkgs=channel:nixos-23.11 -p "$(cat <<EOF
  let
    pkgs = import <nixpkgs> { config = {}; overlays = []; };
    iso-config = pkgs.path + /nixos/modules/installer/cd-dvd/installation-cd-
    nixos = pkgs.nixos iso-config;
  in nixos.config.system.build.nixos-generate-config
EOF
)"
```

It does the following:

- Provide Nixpkgs from a channel
- Take the configuration file for the minimal ISO image from the obtained version of the Nixpkgs repository
- Evaluate that NixOS configuration with `pkgs.nixos`
- Return the derivation which produces the `nixos-generate-config` executable from the evaluated configuration

By default, the generated configuration file is written to `/etc/nixos/configuration.nix`. To avoid overwriting this file you have to specify the output directory. Create a NixOS configuration in your working directory:

```
$ nixos-generate-config --dir ./
```

In the working directory you will then find two files:

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

Skip to main content

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 = "23.11";
}
```

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

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

We add two lightweight programs as an example:

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

NixOS

On NixOS your 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:

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:

```nix
{ 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 = "23.11";
}
```

Creating a QEMU based virtual machine from a NixOS configuration

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

Skip to main content

```
$ nix-build '<nixpkgs/nixos>' -A vm -I nixpkgs=channel:nixos-23.11 -I nixos-config=./c
```

This command builds the attribute `vm` from the `nixos-23.11` release of NixOS, using the NixOS configuration as specified in the relative path.

Detailed explanation

Running the virtual machine

The previous command created a link with the name `result` in the working directory. It links to the directory that contains the virtual machine.

```
$ ls -R ./result
result:
bin  system

result/bin:
run-nixos-vm
```

Run the virtual machine:

```
$ QEMU_KERNEL_PARAMS=console=ttyS0 ./result/bin/run-nixos-vm -nographic; reset
```

This command will run QEMU in the current terminal due to `-nographic`. `console=ttyS0` will also show the boot process, which ends at the console login screen.

Log in as `alice` with the password `test`. Check that the programs are indeed available as specified:

```
$ cowsay hello | lolcat
```

Exit the virtual machine by shutting it down:

```
$ sudo poweroff
```

Skip to main content

Note

If you forgot to add the user to `wheel` or didn't set a password, stop the virtual machine from a different terminal:

```
$ sudo pkill qemu
```

Running the virtual machine will create a `nixos.qcow2` file in the current directory. This disk image file contains the dynamic state of the virtual machine. It can interfere with debugging as it keeps the state of previous runs, for example the user password.

Delete this file when you change the configuration:

```
$ rm nixos.qcow2
```

Running GNOME on a graphical VM

To create a virtual machine with a graphical user interface, add the following lines to the configuration:

```
1  # Enable the X11 windowing system.
2  services.xserver.enable = true;
3
4  # Enable the GNOME Desktop Environment.
5  services.xserver.displayManager.gdm.enable = true;
6  services.xserver.desktopManager.gnome.enable = true;
```

These three lines activate X11, the GDM display manager (to be able to login) and Gnome as desktop manager.

Skip to main content

NixOS

On NixOS, use `installation-cd-graphical-gnome.nix` to generate the configuration file:

```
nix-shell -I nixpkgs=channel:nixos-23.11 -p "$(cat <<EOF
  let
    pkgs = import <nixpkgs> { config = {}; overlays = []; };
    iso-config = pkgs.path + /nixos/modules/installer/cd-dvd/installation-cd-
    nixos = pkgs.nixos iso-config;
  in nixos.config.system.build.nixos-generate-config
EOF
)"
```

```
$ nixos-generate-config --dir ./
```

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

```nix
{ 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 = "23.11";
}
```

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

```
$ nix-build '<nixpkgs/nixos>' -A vm -I nixpkgs=channel:nixos-23.11 -I nixos-config=./c
$ ./result/bin/run-nixos-vm
```

Skip to main content

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:

```nix
{ 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 = "23.11";
}
```

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

References

- NixOS Manual: NixOS Configuration.
- NixOS Manual: Modules.

Skip to main content

- NixOS Manual: Changing the configuration.
- NixOS source code: `configuration template` in `tools.nix`.
- NixOS source code: `vm` attribute in `default.nix`.
- Nix manual: `nix-build`.
- Nix manual: common command-line options.
- Nix manual: `NIX_PATH` environment variable.
- 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