NixOS-WSL

[m] chat 292 users | nixpkgs 24.05 | downloads 42k

Modules for running NixOS on the Windows Subsystem for Linux

Documentation is available here

Quick Start

1. Enable WSL if you haven't done already:

- `wsl --install --no-distribution`

2. Download `nixos-wsl.tar.gz` from the latest release.

3. Import the tarball into WSL:

- `wsl --import NixOS $env:USERPROFILE\NixOS\ nixos-wsl.tar.gz --version 2`

4. You can now run NixOS:

- `wsl -d NixOS`

For more detailed instructions, refer to the documentation.

License

Apache License, Version 2.0. See `LICENSE` or http://www.apache.org/licenses/LICENSE-2.0.html for details.

Installation

System requirements

NixOS-WSL is tested with the Windows Store version of WSL 2, which is now available on all supported Windows releases (both 10 and 11). Support for older "inbox" versions is best-effort.

Install NixOS-WSL

First, download the latest release.

Then open up a PowerShell and run:

```
wsl --import NixOS $env:USERPROFILE\NixOS\ nixos-wsl.tar.gz --version 2
```

Or for Command Prompt:

```
wsl --import NixOS %USERPROFILE%\NixOS\ nixos-wsl.tar.gz --version 2
```

This sets up a new WSL distribution `NixOS` that is installed in a directory called `NixOS` inside your user directory. `nixos-wsl.tar.gz` is the path to the file you downloaded earlier. You can adjust the installation path and distribution name to your liking.

To get a shell in your NixOS environment, use:

```
wsl -d NixOS
```

If you chose a different name for your distro during import, adjust this command accordingly.

Post-Install

After the initial installation, you need to update your channels once, to be able to use `nixos-rebuild`:

```
sudo nix-channel --update
```

If you want to make NixOS your default distribution, you can do so with

```
wsl -s NixOS
```

Design

Getting NixOS to run under WSL requires some workarounds:

- instead of directly loading systemd, we use a small shim that runs the NixOS activation scripts first
- some additional binaries required by WSL's internal tooling are symlinked to FHS paths on activation

Running on older WSL versions also requires a workaround to spawn systemd by hijacking the root shell and spawning a container with systemd inside. This method of running things is deprecated and will be removed with the 24.11 release.

Building your own system tarball

This requires access to a system that already has Nix installed. Please refer to the Nix installation guide if that's not the case.

If you have a flakes-enabled Nix, you can use the following command to build your own tarball instead of relying on a prebuilt one:

```
sudo nix run github:nix-community/NixOS-
WSL#nixosConfigurations.default.config.system.build.tarballBuilder
```

Or, if you want to build with local changes, run inside your checkout:

```
sudo nix run .#nixosConfigurations.your-
hostname.config.system.build.tarballBuilder
```

Without a flakes-enabled Nix, you can build a tarball using:

```
nix-build -A nixosConfigurations.default.config.system.build.tarballBuilder &&
sudo ./result/bin/nixos-wsl-tarball-builder
```

The resulting tarball can then be found under `nixos-wsl.tar.gz`.

How-To

This section contains guides for some common things you might want to do with your NixOS-WSL installation.

Setup VSCode Remote

The VSCode Remote server can not be run as-is on NixOS, because it downloads a nodejs binary that requires `/lib64/ld-linux-x86-64.so.2` to be present, which isn't the case on NixOS.

There are two options to get the server to run. Option 1 is more robust but might impact other programs. Option 2 is a little bit more brittle and sometimes breaks on updates but doesn't influence other programs. Both options require `wget` to be installed:

```
environment.systemPackages = [
    pkgs.wget
];
```

Option 1: Set up nix-ld

nix-ld is a program that provides `/lib64/ld-linux-x86-64.so.2`, allowing foreign binaries to run on NixOS.

Running the VSCode server on NixOS-WSL requires using nix-ld 2.0 which is as of writing only on NixOS unstable or nix-ld-rs on NixOS 24.05.

To set it up, add the following to your configuration:

```
programs.nix-ld = {
    enable = true;
    package = pkgs.nix-ld-rs; # only for NixOS 24.05
};
```

Option 2: Patch the server

The other option is to replace the nodejs binary that ships with the vscode server with one from the nodejs nixpkgs package. This module will set up everything that is required to get it running. If you are using flakes, you can add that repo as a flake input and include it from there. Otherwise, copy the file to your configuration and add it to your imports.

Add the following to your configuration to enable the module:

```
vscode-remote-workaround.enable = true;
```

How to change the username

If you want to change the default username to something other than `nixos`, use the `wsl.defaultUser` option. When building your own tarball, this should be sufficient. A user with the name specified in that option will be created automatically.

Changing the username on an already installed system is possible as well. Follow these instructions to make sure, the change gets applied correctly:

1. Change the `wsl.defaultUser` setting in your configuration to the desired username.
2. Apply the configuration:
   `sudo nixos-rebuild boot`
   Do not use `nixos-rebuild switch`! It may lead to the new user account being misconfigured.
3. Exit the WSL shell and stop your NixOS distro:
   `wsl -t NixOS`.
4. Start a shell inside NixOS and immediately exit it to apply the new generation:
   `wsl -d NixOS --user root exit`
5. Stop the distro again:
   `wsl -t NixOS`
6. Open a WSL shell. Your new username should be applied now!

How to configure NixOS-WSL with flakes

First add a `nixos-wsl` input, then add `nixos-wsl.nixosModules.default` to your nixos configuration.

Below is a minimal `flake.nix` for you to get started:

```
{
  inputs = {
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-unstable";
    nixos-wsl.url = "github:nix-community/NixOS-WSL/main";
  };

  outputs = { self, nixpkgs, nixos-wsl, ... }: {
    nixosConfigurations = {
      nixos = nixpkgs.lib.nixosSystem {
        system = "x86_64-linux";
        modules = [
          nixos-wsl.nixosModules.default
          {
            system.stateVersion = "24.05";
            wsl.enable = true;
          }
        ];
      };
    };
  };
}
```

Install MSIXBundle Certificate

To use the `.msixbundle` launcher some systems need to install the certificate for it. The certificate is included in the launcher and can be accessed from it's properties. The certificate needs to be installed in the `Trusted People` certificate store on the local machine which requires administrator privileges.

Step by step instructions

1. Open `.msixbundle` files **properties**
2. Select **Digital Signatures** tab
3. Select signature named `nzbr`
4. Click **details**
5. Click **View Certificate**
6. Click **Install Certificate**
7. Select `Local Machine` and click **Next**
8. Select `Place all certificates in the following store` and click **Browse**
9. Select `Trusted People` from the list and click **OK**
10. Click **Next** and then **Finish**

You should now be able to use the `.msixbundle` launcher.

Troubleshooting

General Tips

- Try fully restarting WSL by running `wsl --shutdown`. This will close all your terminal windows. Then just restart wsl in your terminal.
  Please keep in mind that this will also end any process you might have running in other WSL distros. If that is currently not an option, you may try `wsl -t nixos`, which will just stop the `nixos` distro. (You may need to change that if you imported the distro under some other name). However, some issues will only be resolved after a *full* restart of WSL.
- Make sure that you are using the Microsoft Store version of WSL
- Update WSL2 to the latest version
  - To update, run: `wsl --update`
  - To check which version you currently have installed, run `wsl --version`
    - The latest version can be found on the Microsoft/WSL repo
    - If this command does not work, you are probably not using the Microsoft Store version of WSL!

Recovery Shell

A recovery shell can be started with

```
wsl -d NixOS --system --user root -- /mnt/wslg/distro/bin/nixos-wsl-recovery
```

This will load the WSL "system" distribution, activate your configuration, then chroot into your NixOS system, similar to what `nixos-enter` would do on a normal NixOS install.

You can choose an older generation to load with

```
wsl -d NixOS --system --user root -- /mnt/wslg/distro/bin/nixos-wsl-recovery --
system /nix/var/nix/profiles/system-42-link
```

(note that the path is relative to the new root)

wsl.enable

Whether to enable support for running NixOS as a WSL distribution.

Type: boolean

Default: `false`

Example: `true`

Declared by:

- <nixos-wsl>/modules/wsl-distro.nix

wsl.defaultUser

The name of the default user

Type: string

Default: `"nixos"`

Declared by:

- <nixos-wsl>/modules/wsl-distro.nix

wsl.docker-desktop.enable

Whether to enable Docker Desktop integration.

Type: boolean

Default: `false`

Example: `true`

Declared by:

- <nixos-wsl>/modules/docker-desktop.nix

wsl.extraBin

Additional files to be added to /bin

Type: list of (submodule)

Declared by:

- <nixos-wsl>/modules/wsl-distro.nix

wsl.extraBin.*.copy

Whether or not the file should be copied instead of symlinked

Type: boolean

Default: `false`

Declared by:

- <nixos-wsl>/modules/wsl-distro.nix

wsl.extraBin.*.name

The name the file should be created as in /bin

Type: string

Default: `baseNameOf src`

Declared by:

- <nixos-wsl>/modules/wsl-distro.nix

wsl.extraBin.*.src

Path of the file that should be added

Type: string

Declared by:

- <nixos-wsl>/modules/wsl-distro.nix

wsl.interop.includePath

Include Windows PATH in WSL PATH

Type: boolean

Default: `true`

Declared by:

- <nixos-wsl>/modules/interop.nix

wsl.interop.register

Explicitly register the binfmt_misc handler for Windows executables

Type: boolean

Default: `false`

Declared by:

- <nixos-wsl>/modules/interop.nix

wsl.nativeSystemd

Use native WSL systemd support

Type: boolean

Default: `true`

Declared by:

- <nixos-wsl>/modules/systemd

wsl.startMenuLaunchers

Whether to enable shortcuts for GUI applications in the windows start menu.

Type: boolean

Default: `false`

Example: `true`

Declared by:

- <nixos-wsl>/modules/wsl-distro.nix

wsl.tarball.configPath

Path to system configuration which is copied into the tarball

Type: null or path

Default: `null`

Declared by:

- <nixos-wsl>/modules/build-tarball.nix

wsl.usbip.enable

Whether to enable USB/IP integration.

Type: boolean

Default: `false`

Example: `true`

Declared by:

- <nixos-wsl>/modules/usbip.nix

wsl.usbip.autoAttach

Auto attach devices with provided Bus IDs.

Type: list of string

Default: `[ ]`

Example:

```
[
  "4-1"
]
```

Declared by:

- <nixos-wsl>/modules/usbip.nix

wsl.usbip.snippetIpAddress

This snippet is used to obtain the address of the Windows host where Usbipd is running. It can also be a plain IP address in case networkingMode=mirrored or wsl-vpnkit is used.

Type: string

Default: `"$(ip route list | sed -nE 's/(default)? via (.*) dev eth0 .*/\\2/p' | head -n1)"`

Example: `"127.0.0.1"`

Declared by:

- <nixos-wsl>/modules/usbip.nix

wsl.useWindowsDriver

Whether to enable OpenGL driver from the Windows host.

Type: boolean

Default: `false`

Example: `true`

Declared by:

- <nixos-wsl>/modules/wsl-distro.nix

wsl.wrapBinSh

Wrap /bin/sh with a script that sets the correct environment variables (like the user shells). Only takes effect when using native systemd

Type: boolean

Default: `true`

Declared by:

- <nixos-wsl>/modules/systemd/native/wrap-shell.nix

wsl.wslConf

Configuration values for /etc/wsl.conf. See https://learn.microsoft.com/en-us/windows/wsl/wsl-config#configuration-settings-for-wslconf for all options supported by WSL.

Type: attribute set of section of an INI file (attrs of INI atom (null, bool, int, float or string))

Default: `{ }`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.automount.enabled

Automatically mount windows drives under /mnt

Type: boolean

Default: `true`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.automount.ldconfig

Wether to modify `/etc/ld.so.conf.d/ld.wsl.conf` to load OpenGL drivers provided by the Windows host in `/usr/lib/wsl/lib` with `/sbin/ldconfig`. This way of providing OpenGL drivers does not work with NixOS and `wsl.useWindowsDriver` should be used instead.

Type: boolean

Default: `false`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.automount.mountFsTab

Mount entries from /etc/fstab through WSL. You should probably leave this on false, because systemd will mount those for you.

Type: boolean

Default: `false`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.automount.options

Comma-separated list of mount options that should be used for mounting windows drives.

Type: strings concatenated with “,”

Default: `"metadata,uid=1000,gid=100"`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.automount.root

The directory under which to mount windows drives.

Type: string matching the pattern ^/.*[^/]$

Default: `"/mnt"`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.boot.command

A command to run when the distro is started.

Type: string

Default: `""`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.boot.systemd

Use systemd as init. There's no need to enable this manually, use the wsl.nativeSystemd option instead

Type: boolean

Default: `false`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.interop.enabled

Support running Windows binaries from the linux shell.

Type: boolean

Default: `true`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.interop.appendWindowsPath

Include the Windows PATH in the PATH variable

Type: boolean

Default: `true`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.network.generateHosts

Generate /etc/hosts through WSL

Type: boolean

Default: `true`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.network.generateResolvConf

Generate /etc/resolv.conf through WSL

Type: boolean

Default: `true`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.network.hostname

The hostname of the WSL instance

Type: string

Default: `"config.networking.hostName"`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix

wsl.wslConf.user.default

Which user to start commands in this WSL distro as

Type: string

Default: `"root"`

Declared by:

- <nixos-wsl>/modules/wsl-conf.nix