MENU

Mattia Gheda

Thoughts on Software, NixOS, Elixir and self discovery

Tutorial: Getting started with Home Manager for Nix

April 5, 2021

My opinion is that Home Manager **provides a radically better way to manage a user's environment** for both packages and dotfiles, effectively allowing you to take a **configuration as code** approach.

The **goal of this post** is to show you how to **get started with Home Manager** from scratch, install packages, port over some existing configuration and explore more advanced features.

This content does not require nor assumes knowledge of the **Nix language**, although you might find helpful to read some of the resources linked at the end of the post.

> If you want to know more about Nix you can also read this post

Importantly by following this strategy you will be able to **port your existing configuration incrementally** and do it at your own pace.

Introduction

One of the things that I've always found cumbersome of setting up a new laptop/desktop/server has been managing my user-level configuration and packages. I'm referring to things like `git` , `vim` , `tmux` , `ssh` , window manager and terminal settings for my desktop machines and in general applications that are only required for my current user.

Over the years I tried several solutions for my configuration:

- do nothing and start with a clean slate every time
- copy over files
- a `dotfiles` repository, sometimes managed with bash scripts like homeshick

For apps management I've used Homebrew on MacOS or my distro's package manager and always installed packages globally.

After moving to NixOS I've finally landed on **the Nix solution to this problem: Home Manager**.

Home Manager allows you to use Nix's declarative approach to manage your user-level configuration and packages. It works on any *nix system supported by Nix, including MacOS.

Prerequisites

To get started you'll need two things: Nix and Home Manager.

First, install Nix

If you are not on NixOS the first thing you need to install is the Nix package manager.

Run the following command (you will need `sudo` access).

```
curl -L https://nixos.org/nix/install | sh
```

Once complete follow the instructions to load `nix` in the current shell or restart your terminal.

Test that everything worked by typing.

```
nix
```

You should see a help message.

Install Home Manager

Detailed instructions (and warnings) are available on the Home Manager homepage.

If you followed the instructions above to install `nix` you will now be on the `unstable` channel of the nix package tree (the one that has the most up to date packages).

The next step then is to add the appropriate channel for Home Manager.

```
nix-channel --add https://github.com/nix-community/home-manager/archive/master.tar.gz home-mana
nix-channel --update
```

Change the command above as required if you are tracking a different nix channel.

Now we can install `home-manager` by typing

```
nix-shell '<home-manager>' -A install
```

You can test that everything worked by typing

```
home-manager
```

Congratulations, you are ready to go!

Getting started

As you have seen in the install prompt, by default Home Manager initializes a new configuration in

```
$HOME/.config/nixpkgs/home.nix
```

that should look roughly like this

```
{ config, pkgs, ... }:

{
  # Let Home Manager install and manage itself.
  programs.home-manager.enable = true;

  # Home Manager needs a bit of information about you and the
  # paths it should manage.
  home.username = "ghedamat";
  home.homeDirectory = "/home/ghedamat";

  # This value determines the Home Manager release that your
  # configuration is compatible with. This helps avoid breakage
  # when a new Home Manager release introduces backwards
  # incompatible changes.
  #
  # You can update Home Manager without changing this value. See
  # the Home Manager release notes for a list of state version
  # changes in each release.
```

```
    home.stateVersion = "21.05";
  }
```

Git-it

The first step is to move this configuration to a git repository, I prefer to have it in a different location and use symlinks to expose it to Home Manager.

Run:

```
mv ~/.config/nixpkgs ~/nixfiles
cd ~/nixfiles
git init
git add .
git commit -m 'getting started with Home Manager'
cd ~/.config
ln -s ~/nixfiles nixpkgs
```

Then test that you can apply the configuration.

```
home-manager switch
```

We have not added anything so this should be a no-op.

Home Manager will let you know that it's "reusing lastest profile generation".

Let's add our first package

First let's see how we can use Home Manager to install packages for our user. In this example we'll add `tmux`. Edit `~/nixfiles/home.nix` as follows:

```
   home.homeDirectory = "/home/ghedamat";

+  # Packages to install
+  home.packages = [
+    # pkgs is the set of all packages in the default home.nix implementation
+    pkgs.tmux
+  ];
+
   # This value determines the Home Manager release that your
```

Then run

```
home-manager switch
```

And now try it out

```
tmux
```

A great place to search for packages is https://search.nixos.org/packages, make sure you pick the right "channel", if you are following this guide it will be `unstable` .

What is this `home-manager switch` business?

`home-manager switch` is how you "activate" your configuration. Home Manager will evaluate your configuration, build the result and **atomically** switch your old configuration with the new one.

This also means that it's possible to see all old configurations

```
$ home-manager generations
2021-04-03 01:39 : id 2 -> /nix/store/5v5gnkq7ddkikdpghxlp039zfzxib1x1-home-manager-generation
2021-04-02 21:41 : id 1 -> /nix/store/rjzjszmwfrhmwzvqxhgy4l2a4rrr2xma-home-manager-generation
```

And you can rollback to older versions as well.

```
# copy the path from the command above and add /activate
 /nix/store/rjzjszmwfrhmwzvqxhgy4l2a4rrr2xma-home-manager-generation/activate
 ...
 # this will create a new generation
 Creating profile generation 3
 ...
```

You can then switch again to re-apply your changes to go back to the current version of `home.nix` .

```
$ home-manager switch
```

Porting over an existing `dotfile`

So far we saw how to use Home Manager to install packages for our user, but a perhaps more important use case is manage our user configuration.

First we'll see how to take an existing configuration file and make it part of Home Manager.

The simplest way to do this is to use the `home.file` option.

Assume that you have a `~/.vimrc` with the following contents:

```
call plug#begin()
Plug 'LnL7/vim-nix'
```

```
call plug#end()
```

First let’s move it in our nixfiles repo

```
mv ~/.vimrc ~/nixfiles/vimrc
```

You can then edit `~/nixfiles/home.nix` as follows

```
      pkgs.tmux
    ];

+   # Raw configuration files
+   home.file.".vimrc".source = ./vimrc;
+
    # This value determines the Home Manager release that your
```

And run `home-manager switch` again.

Now, let’s check what happened

```
$ ls -l ~/.vimrc
lrwxrwxrwx 1 ghedamat ghedamat 69 Apr  3 05:38 /home/ghedamat/.vimrc -> /nix/store/ynhkrdrc7hz
```

`/.vimrc` is now a symlink to a file in the “Nix store”, the place where all the nix things go. Without concering ourself with details, the thing to notice is that if you now change the contents of `~/nixfiles/vimrc` and re-run `home-manager switch` Home Manager will detect the changes, create a new version of `.vimrc` in the Nix store and update the symlink.

```
$ echo "hello nix" > ~/nixfiles/vimrc
$ home-manager switch
$ ls -l ~/.vimrc
lrwxrwxrwx 1 ghedamat ghedamat 69 Apr  3 05:47 /home/ghedamat/.vimrc -> /nix/store/dsq0da2y4p7
```

It is true that managing configuration in this way **will add a step** every time you want to edit your `vimrc` . I believe that this **tradeoff is worth it even if you were to decide to not use any other feature offered by Home Manager** as you now have a reliable and consistent way to manage all your configuration files and packages.

Using Home Manager modules

Using the `home.file` configuration option is my preferred way to port existing configuration files . Once that’s done though **Home Manager has much more to offer**.

Home Manager comes with a large amount of pre-defined configuration `modules` for a variety of applications (full list on github). These modules allow you to use a consistent syntax (the Nix language) to configure every application regardless of the output format that each program requires (ini, yaml, custom…).

By using modules you will get **type safety guarantees** about your configuration as it will be now written in Nix and the modules can specify types for each option you pass. This also means that **you can use the Nix language to add logic** (i.e conditionals and functions) as well as the ability to compose your configuration as you would with any other functional program written in Nix.

The downside is that you have to learn at least a *small part of the Nix language* (mostly how to write `sets` , which are similar to `maps` and `hash` in other languages).

Once you have identified a module you are interested in, all the options available are listed in the Home Manager Manual.

Porting your `git` config

Let's see an example with porting over our `~/.config/git/config` to Home Manager.

```
# current contents of ~/.config/git/config
[user]
        email = ghedamat@gmail.com
        name = ghedamat
[alias]
        st = "status"
```

Edit `home.nix` as follows (you can find the full list of options for `programs.git` here)

```
   home.file.".vimrc".source = ./vimrc;

+  # Git config using Home Manager modules
+  programs.git = {
+    enable = true;
+    userName = "ghedamat";
+    userEmail = "ghedamat@gmail.com";
+    aliases = {
+      st = "status";
+    };
+  };
+
   # This value determines the Home Manager release that your
```

Let's try to apply this change

```
$ home-manager switch
Existing file '/home/ghedamat/.config/git/config' is in the way of '/nix/store/9k4rq2d247qcs9n
```

```
Please move the above files and try again or use -b <ext> to move automatically.
```

Ah! Home Manager noticed that we already have that file in the system and will **not** override it with the one it generates. Neat!

The fix is simple:

```
rm ~/.config/git/config
home-manager switch
```

We can verify that the file is now generated by Home Manager (notice the content is slightly different)

```
$ cat ~/.config/git/config
[alias]
        st = "status"

[user]
        email = "ghedamat@gmail.com"
        name = "ghedamat"
```

Structuring your Home Manager config

The author of Home Manager recommends to **start with a single `home.nix` file** and I would definitely agree. As you learn more about the Nix language you'll find about all the different ways to structure your code.

Later, you might want to learn about using imports to break down your configuration into multiple files. A more advance approach is to build your own Nix modules.

I might decide to cover these in a future post.

Conclusion

There is much more to discuss and cover but this should be enough for a short introductory tutorial that aims at getting you started.

In summary:

- Start simple! Install Home Manager and **port over a few files at the time**
- Start by using `home.file` for most things
- Move over to using Home Manager modules as needed and/or try to use them for new configurations
- Explore the resources below to learn about Nix and see what others are doing with Home Manager

I also want to leave a big **thank you** to the Home Manager contributors and the Nix community at large, these tools have greatly improved my workflow.

Let me know what you think!

Feel free to reach out with **any comments/questions/suggestions and correction** about what's written in this article.

My hope is to see more people adopting Home Manager and Nix :)

Testimonials

> "Let me tell you how much I dislike configuring dev machines: I use Macs but haven't installed dev tools since 2011. I got tired of OS X releases breaking them so I do all of my dev on external machines or VMs. I tried Home Manager and it ended working better than I expected." @elucid

> "I see how it is a better idea that just hand configuring everything poorly like I do now." @benjamintmoss

Resources

Here's a list of other blog posts and resources that will help you get started:

- Home Manager github page
- Home Manager manual
- Original Home Manager announcement
- Homebrew to Home Manager
- Your home in Nix
- Interesting template to get started with Home Manager
- Burke Libbey Nixology series (4 videos related to Home Manager)
- Primer on Nix terminology
- An introduction to the Nix ecosystem

Many thanks to @shazow and @eviltrout for their feedback on this article.

Previous
← *A Client-Server development environment*

Next
Using niv to install recent Elixir in your nix-shell →