User Program and Configuration Management with `home-manager`

OtaNix Workshop

Niklas Halonen, Joonas von Lerber

2025-01-22

Aalto University

Agenda

1. Introduction and basics
2. Hands-on installation to a VM
   - ❄ Follow along!


SOCIETY IF
NIX USERS STOPPED SPENDING ALL THEIR TIME CONFIGURING THEIR SYSTEM
imgflip.com

What is home-manager (H-M)?

1. A Nix module for managing user applications and services, and their configuration, a.k.a *dotfiles*.
2. A CLI for interacting and invoking the H-M module.

Home-manager's (mostly a reiteration of Nix's) philosophy:

- Reproducibility: building a configuration leads to a *unique* outcome.
- Separation of concerns: enables splitting code into modules and files.
- Declarative unified[1] configuration.
- Cross reference/link configuration options and variables.
  - Even integrate to the NixOS configuration.
- As always, everything is just a **symbolic link** to the Nix store.

[1] Some H-M modules just provide a `configFile` option, whereas some have more complex `settings` as well as a `configFile`.

A word of warning

- Many modules/services are available on both NixOS and H-M which may conflict with each other if enabled and may have incompatible configuration options or varying feature support.

Installation (standalone)

1. Installing nix and git (if not already installed)
2. Starting a shell with `home-manager` CLI
3. Creating a standalone H-M config repository
4. H-M basics and solving the common OpenGL problem
    - User programs
    - User services
    - Window manager
5. Setting up more complicated H-M integration with FireFox, VSCode

Installing Nix

https://docs.determinate.systems/getting-started/

```
curl -fsSL https://install.determinate.systems/nix | \
    sh -s -- install --determinate
```

Creating a standalone H-M config repository

```
git init ~/dotfiles
nix run home-manager/release-24.11 -- init ~/dotfiles
# Note: home-manager/master is the latest unstable version of H-M
cd ~/dotfiles
git add . # Required because flakes ignore files outside of git
nix run home-manager/release-24.11 -- switch --flake ~/dotfiles
```

Expected outcomes:

- (The news are shown)
- You have the `home-manager` program available
- `dotfiles` repository contains the following files
  - dotfiles
    - .git/
    - flake.nix
    - flake.lock
    - home.nix

Decrypting the Default Configuration

The default `flake.nix` is as follows and is all set to start using H-M so you **don't need to understand** any of it right now:

```
{
  description = "H-M configuration of otanix";

  inputs = {
    # Specify the source of H-M and Nixpkgs.
    nixpkgs.url = "github:nixos/nixpkgs/nixos-
unstable";
    home-manager = {
      url = "github:nix-community/home-manager";
      inputs.nixpkgs.follows = "nixpkgs";
    };
  };

  outputs = { nixpkgs, home-manager, ... }:
    let
      system = "x86_64-linux";
      pkgs = nixpkgs.legacyPackages.${system};
    in {
      homeConfigurations."otanix" =
        home-manager.lib
          .homeManagerConfiguration {
        inherit pkgs;

        # Specify your home configuration
        # modules here, for example,
        # the path to your home.nix.
        modules = [ ./home.nix ];

        # Optionally use extraSpecialArgs
        # to pass through arguments to home.nix
      };
    };
}
```

Decrypting `home.nix`

This is more relevant for day-to-day configuration of H-M.

```nix
{pkgs, ...}:

let
  # Personal Info
  name = "Matti Meikäläinen";
  email = "matti.meikalainen@iki.fi";
  username = "leet-matti";
  githubUsername = "MattimusUltimatus";

  homeDir = "/home/${username}"
in {
  programs = {
    home-manager.enable = true;
    git = {
      enable = true;
      userName = "${name}";
      userEmail = "${email}";
    }
    fish = {
      enable = true;
      shellAbbrs = {
        "l" = "ls -arthl";
      }
    }
  }
}
```

How to Install Programs

Under the `programs` attribute set, you can add programs and configure them. I want to have Firefox so let's add it.

```
programs = {
  firefox = {
    enable = true;
  }
}
```

Rebuilding the Configuration

1. Make changes to the configuration
2. Git add them
3. Run

   ```
   home-manager switch --flake ~/dotfiles
   ```
4. If there's an error
   - Then: decrypt the error message
   - Else: test out the changes
5. Go back to step 1.

H-M Commands

Some of the useful commands provided by `home-manager --help`:

```
Commands
  option OPTION.NAME      Inspect configuration option named OPTION.NAME.

  build                   Build configuration into result directory

  switch                  Build and activate configuration

  generations             List all home environment generations

  packages                List all packages installed in home-manager-path

  uninstall               Remove Home Manager
```

Resources

Here are some useful resources for finding H-M

- https://nix-community.github.io/home-manager/index.xhtml#sec-flakes-standalone
- https://home-manager-options.extranix.com/