Getting Started with Home Manager

As I mentioned earlier, NixOS can only manage system-level configuration. To manage user-level configuration in the Home directory, we need to install Home Manager.

According to the official Home Manager Manual, to install Home Manager as a module of NixOS, we first need to create `/etc/nixos/home.nix`. Here's an example of its contents:

nix

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

{
  # TODO please change the username & home directory to your own
  home.username = "ryan";
  home.homeDirectory = "/home/ryan";

  # link the configuration file in current directory to the specified location i
  # home.file.".config/i3/wallpaper.jpg".source = ./wallpaper.jpg;

  # link all files in `./scripts` to `~/.config/i3/scripts`
  # home.file.".config/i3/scripts" = {
  #   source = ./scripts;
  #   recursive = true;   # link recursively
  #   executable = true;  # make all files executable
  # };

  # encode the file content in nix configuration file directly
  # home.file.".xxx".text = ''
  #     xxx
  # '';

  # set cursor size and dpi for 4k monitor
  xresources.properties = {
    "Xcursor.size" = 16;
    "Xft.dpi" = 172;
  };

  # Packages that should be installed to the user profile.
  home.packages = with pkgs; [
    # here is some command line tools I use frequently
    # feel free to add your own or remove some of them

```

```
    neofetch
    nnn # terminal file manager

    # archives
    zip
    xz
    unzip
    p7zip

    # utils
    ripgrep # recursively searches directories for a regex pattern
    jq # A lightweight and flexible command-line JSON processor
    yq-go # yaml processor https://github.com/mikefarah/yq
    eza # A modern replacement for ‘ls’
    fzf # A command-line fuzzy finder

    # networking tools
    mtr # A network diagnostic tool
    iperf3
    dnsutils  # `dig` + `nslookup`
    ldns # replacement of `dig`, it provide the command `drill`
    aria2 # A lightweight multi-protocol & multi-source command-line download ut
    socat # replacement of openbsd-netcat
    nmap # A utility for network discovery and security auditing
    ipcalc  # it is a calculator for the IPv4/v6 addresses

    # misc
    cowsay
    file
    which
    tree
    gnused
    gnutar
    gawk
    zstd
    gnupg

    # nix related
    #
    # it provides the command `nom` works just like `nix`
    # with more details log output
    nix-output-monitor

    # productivity
```

```
    hugo # static site generator
    glow # markdown previewer in terminal

    btop  # replacement of htop/nmon
    iotop # io monitoring
    iftop # network monitoring

    # system call monitoring
    strace # system call monitoring
    ltrace # library call monitoring
    lsof # list open files

    # system tools
    sysstat
    lm_sensors # for `sensors` command
    ethtool
    pciutils # lspci
    usbutils # lsusb
  ];

  # basic configuration of git, please change to your own
  programs.git = {
    enable = true;
    userName = "Ryan Yin";
    userEmail = "xiaoyin_c@qq.com";
  };

  # starship - an customizable prompt for any shell
  programs.starship = {
    enable = true;
    # custom settings
    settings = {
      add_newline = false;
      aws.disabled = true;
      gcloud.disabled = true;
      line_break.disabled = true;
    };
  };

  # alacritty - a cross-platform, GPU-accelerated terminal emulator
  programs.alacritty = {
    enable = true;
    # custom settings
    settings = {
```

```
      env.TERM = "xterm-256color";
      font = {
        size = 12;
        draw_bold_text_with_bright_colors = true;
      };
      scrolling.multiplier = 5;
      selection.save_to_clipboard = true;
    };
  };

  programs.bash = {
    enable = true;
    enableCompletion = true;
    # TODO add your custom bashrc here
    bashrcExtra = ''
      export PATH="$PATH:$HOME/bin:$HOME/.local/bin:$HOME/go/bin"
    '';

    # set some aliases, feel free to add more or remove some
    shellAliases = {
      k = "kubectl";
      urldecode = "python3 -c 'import sys, urllib.parse as ul; print(ul.unquote_
      urlencode = "python3 -c 'import sys, urllib.parse as ul; print(ul.quote_pl
    };
  };

  # 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 = "24.11";

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

After adding `/etc/nixos/home.nix`, you need to import this new configuration file in `/etc/nixos/flake.nix` to make use of it, use the following command to generate an

example in the current folder for reference:

```shell
nix flake new example -t github:nix-community/home-manager#nixos
```

After adjusting the parameters, the content of `/etc/nixos/flake.nix` is as follows:

```nix
{
  description = "NixOS configuration";

  inputs = {
    nixpkgs.url = "github:nixos/nixpkgs/nixos-24.11";
    # home-manager, used for managing user configuration
    home-manager = {
      url = "github:nix-community/home-manager/release-24.11";
      # The `follows` keyword in inputs is used for inheritance.
      # Here, `inputs.nixpkgs` of home-manager is kept consistent with
      # the `inputs.nixpkgs` of the current flake,
      # to avoid problems caused by different versions of nixpkgs.
      inputs.nixpkgs.follows = "nixpkgs";
    };
  };

  outputs = inputs@{ nixpkgs, home-manager, ... }: {
    nixosConfigurations = {
      # TODO please change the hostname to your own
      my-nixos = nixpkgs.lib.nixosSystem {
        system = "x86_64-linux";
        modules = [
          ./configuration.nix

          # make home-manager as a module of nixos
          # so that home-manager configuration will be deployed automatically wh
          home-manager.nixosModules.home-manager
          {
            home-manager.useGlobalPkgs = true;
            home-manager.useUserPackages = true;

            # TODO replace ryan with your own username
            home-manager.users.ryan = import ./home.nix;

            # Optionally, use home-manager.extraSpecialArgs to pass arguments to
          }
```

```
          ];
        };
      };
    };
  }
```

Then run `sudo nixos-rebuild switch` to apply the configuration, and home-manager will be installed automatically.

> If your system's hostname is not `my-nixos`, you need to modify the name of `nixosConfigurations` in `flake.nix`, or use `--flake /etc/nixos#my-nixos` to specify the configuration name.

After the installation, all user-level packages and configuration can be managed through `/etc/nixos/home.nix`. When running `sudo nixos-rebuild switch`, the configuration of home-manager will be applied automatically. (**It's not necessary to run `home-manager switch` manually**!)

To find the options we can use in `home.nix`, referring to the following documents:

- Home Manager - Appendix A. Configuration Options: A list of all options, it is recommended to search for keywords in it.
  - Home Manager Option Search is another option search tool with better UI.
- home-manager: Some options are not listed in the official documentation, or the documentation is not clear enough, you can directly search and read the corresponding source code in this home-manager repo.

Home Manager vs NixOS

There are many software packages or configurations that can be set up using either NixOS Modules (`configuration.nix`) or Home Manager (`home.nix`), which brings about a choice dilemma: **What is the difference between placing software packages or configuration files in NixOS Modules versus Home Manager, and how should one make a decision?**

First, let's look at the differences: Software packages and configuration files installed via NixOS Modules are global to the entire system. Global configurations are usually stored in

`/etc` , and system-wide installed software is accessible in any user environment.

On the other hand, configurations and software installed via Home Manager will be linked to the respective user's Home directory. The software installed is only available in the corresponding user environment, and it becomes unusable when switched to another user.

Based on these characteristics, the general recommended usage is:

- NixOS Modules: Install system core components and other software packages or configurations needed by all users.
  - For instance, if you want a software package to continue working when you switch to the root user, or if you want a configuration to apply system-wide, you should install it using NixOS Modules.
- Home Manager: Use Home Manager for all other configurations and software.

The benefits of this approach are:

1. Software and background services installed at the system level often run with root privileges. Avoiding unnecessary software installations at the system level can reduce the security risks of the system.
2. Many configurations in Home Manager are universal for NixOS, macOS, and other Linux distributions. Choosing Home Manager to install software and configure systems can improve the portability of configurations.
3. If you need multi-user support, software and configurations installed via Home Manager can better isolate different user environments, preventing configuration and software version conflicts between users.

How to use packages installed by Home Manager with privileged access?

The first thing that comes to mind is to switch to `root` , but then any packages installed by the current user through `home.nix` will be unavailable. let's take `kubectl` as an example(it's pre-installed via `home.nix`):

sh

```
# 1. kubectl is available
› kubectl | head
kubectl controls the Kubernetes cluster manager.

```

```
 Find more information at: https://kubernetes.io/docs/reference/kubectl/
......

# 2. switch user to `root`
› sudo su

# 3. kubectl is no longer available
> kubectl
Error: nu::shell::external_command

  × External command failed
   ╭─[entry #1:1:1]
 1 │ kubectl
   · ──┬───
   ·   └── executable was not found
   ╰────
  help: No such file or directory (os error 2)


/home/ryan/nix-config> exit
```

The solution is to use `sudo` to run the command, which temporarily grants the current user the ability to run the command as a privileged user (`root`):

sh

```
› sudo kubectl
kubectl controls the Kubernetes cluster manager.
...
```

Loading comments...