timokau / **dotfiles** Public

Code Issues Pull requests Actions Projects Wiki Security I

dotfiles / nixos / **configuration.nix**

timokau Remove sound option ef8aad3 · 6 months ago

379 lines (328 loc) · 9.63 KB

Code Blame Raw

```
1   { config
2   , pkgs # nixpkgs is pinned
3   , ...
4   }:
5
6   let
7     # This is done implicitly on the second run by setting nixPath. Doing it
8     # directly leads to quirky behaviour because the NixOS module would still
9     # come from the old nixpkgs.
10    # pkgs = import (import ./nixpkgs.nix) {};
11    server_address = builtins.readFile ./server_address; # not version controlled
12
13    wireguard = {
14      port = 51822;
15      publicKey = {
16        server = "MRA6FjAwPViS/qsAOpa/eAbeMuHcal6zt/8m4u4hI0w=";
17        pad = "YoUI02AyBRNM7//UTzUlO90mCx7wHX+Jzxf2uaFR3gg=";
18        desk = "d5KwIeKll+z5ZyAVRotC69RXuwM4VLwNtZoRoQEbTjo=";
19      };
20      ip = {
21        server = "10.10.10.1";
22        pad = "10.10.10.2";
23        desk = "10.10.10.3";
24        phone = "10.10.10.4";
25        yoga = "10.10.10.5";
26      };
27    };
28
29    inherit (pkgs) lib;
30  in
31  {
32    imports = [
33      ./hardware-configuration.nix
34      # Different file for each host. Symlink one of the files in `hosts`, e.g.
35      # `ln -s hosts/desk.nix host.nix`. The symlink is not version controlled.
36      # Needs to set `networking.hostName` and `system.stateVersion`.
```

```
    ./host.nix
  ];

  services.autorandr.enable = true;

  # Firmware updates
  services.fwupd.enable = true;

  services.printing = {
    enable = true;
  };

  # Load the i2c module, grant access to users in the i2c group and users with
  # a seat. This is required by ddccontrol.
  hardware.i2c.enable = true;
  # https://github.com/jonls/redshift/issues/436
  # Control monitor brightness, useful for redshift hooks on the user level.
  services.ddccontrol.enable = true;

  # Run fstrim weekly to maintain SSD performance
  services.fstrim = {
    enable = true;
    interval = "weekly";
  };

  nix = {
    settings.sandbox = true;
    nixPath = [
      # Fix the nixpkgs this configuration was built with. To switch to a new
      # revision, explicitly pass it through NIX_PATH once and then it will be
      # set as the new default.
      "nixpkgs=/run/current-system/nixpkgs"
      "nixos-config=/etc/nixos/configuration.nix"
    ];
  };
  # downgrading to read lock on '/nix/var/nix/temproots/18942'
  # copied source '/nix/store/azqqifyxvlgf48lgqh7zmyj0f4az03v9-nixpkgs-e89b21504f3e61e5352
  # acquiring write lock on '/nix/var/nix/temproots/18942
  system.extraSystemBuilderCmds = let
    # make sure store paths are not copied to the store again, which leads to
    # long filenames (https://github.com/NixOS/nix/issues/1728)
    nixpkgs_str = if lib.isStorePath pkgs.path then builtins.storePath pkgs.path else pkgs
  in ''
    ln -sv '${nixpkgs_str}' "$out/nixpkgs"
    echo '${pkgs.path}'
  '';

  # install man pages
  environment.extraOutputsToInstall = [ "man" ];

  # only some administrative packages are installed at the system level
  environment.systemPackages = (with pkgs; [
    man-pages
```

```
90          # android-udev-rules
91          # noto-fonts
92          # dhcpcd
93          acpi
94          gnupg
95          psmisc # killall
96          git
97          vim
98          ranger
99          tree
100         htop
101         rsync
102         ripgrep
103         home-manager # manage user configurations
104         virt-manager
105       ]);
106
107       # disable system sounds
108       xdg.sounds.enable = false;
109
110
111       # firejail needs to run setuid
112       security.wrappers.firejail = {
113         program = "firejail";
114         source = "${pkgs.firejail.out}/bin/firejail";
115         owner = "root";
116         group = "root";
117         setuid = true;
118         setgid = true;
119       };
120
121       programs.firejail = {
122         enable = true;
123         wrappedBinaries = {
124           anki = {
125             executable = "${lib.getBin pkgs.anki}/bin/anki";
126             profile = "${pkgs.firejail}/etc/firejail/anki.profile";
127           };
128         };
129       };
130
131       programs.adb.enable = true;
132       # programs.command-not-found.enable = true;
133
134       fonts = {
135         enableDefaultPackages = true;
136         packages = with pkgs; [
137           source-code-pro
138           inconsolata
139           terminus_font
140           inter # Used in the emacs config
141         ];
```

```
142       };
143
144       # Use the systemd-boot EFI boot loader.
145       boot.loader = {
146         systemd-boot.enable = true;
147         efi.canTouchEfiVariables = true;
148       };
149
150       boot.supportedFilesystems = [ "ntfs" ];
151
152       boot.kernel.sysctl = {
153         # https://wiki.archlinux.org/index.php/zswap
154         "zswap.enabled" = 1;
155         "kernel.sysrq" = 1; # enable "magic sysrq" to force OOM reaper
156       };
157
158       # My laptop freezes at boot with Linux 6.6, so avoid latest for now.
159       # boot.kernelPackages = pkgs.linuxPackages_latest;
160       boot.kernelPackages = pkgs.linuxPackages;
161
162       boot.tmp.cleanOnBoot = true;
163
164       # Container runtime & builder, needs subuids and subgids
165       virtualisation.podman = {
166         enable = true;
167       };
168
169       # Needed to use virt-manager
170       virtualisation.libvirtd.enable = true;
171       programs.dconf.enable = true;
172
173       services.openssh = {
174         enable = true;
175         settings = {
176           PasswordAuthentication = false;
177           PermitRootLogin = "no";
178         };
179         ports = [ 2143 ];
180       };
181
182       # internationalisation properties
183       i18n.defaultLocale = "en_US.UTF-8";
184       console.keyMap = "de";
185
186       time.timeZone = "Europe/Berlin";
187
188       networking = {
189         # use cloudflare dns which is uncensored (in contrast to that of my isp)
190         nameservers = [ "1.1.1.1" ];
191         networkmanager.insertNameservers = [ "1.1.1.1" ];
192
193         # use networkmanager for easy wifi setup
194         networkmanager.enable = true;
```

```
195
196         # block all non-whitelisted connections
197         firewall = {
198           enable = true;
199           allowedTCPPorts = [
200             22000 # syncthing sharing
201             8200 # proxy
202           ];
203           allowedUDPPorts = [
204             21027 # syncthing discovery
205             wireguard.port
206             22
207             8200 # proxy
208           ];
209         };
210       };
211
212       powerManagement = {
213         # support suspend-to-ram, save power
214         enable = true;
215
216         # log boots and wakes from suspend
217         powerUpCommands = "date -Ih >> /var/log/power_up.log";
218       };
219
220       services.snapper = {
221         snapshotInterval = "hourly";
222         snapshotRootOnBoot = true;
223         configs = {
224           root = {
225             SUBVOLUME = "/";
226             TIMELINE_CREATE = true;
227             TIMELINE_CLEANUP = true;
228           };
229           persist = {
230             SUBVOLUME = "/home/timo/p";
231             TIMELINE_CREATE = true;
232             TIMELINE_CLEANUP = true;
233           };
234         };
235       };
236
237       networking.hosts = {
238         # give names to devices in my home network
239         "192.168.0.22" = [ "desk-local" ];
240         "${wireguard.ip.desk}" = [ "desk" ];
241         "${wireguard.ip.server}" = [ "server" ];
242         "${wireguard.ip.pad}" = [ "pad" ];
243         "192.168.0.21" = [ "opo" ];
244         "192.168.0.20" = [ "laptop" ];
245         "192.168.0.26" = [ "kindle" ];
246         "192.168.0.45" = [ "par" ];
```

```
247         "192.168.0.1" = [ "rooter" ];
248         "192.168.0.100" = [ "eb" ];
249       };
250
251       services.xserver = {
252         # Enable the X11 windowing system.
253         enable = true;
254         layout = "de";
255         xkbOptions = "eurosign:e";
256         displayManager = {
257           startx.enable = true;
258           # job.preStart = "${pkgs.xorg.setxkbmap}/bin/setxkbmap de";
259           sessionCommands = "/home/timo/.xprofile"; # TODO setupCommands?
260         };
261       };
262
263       environment.shells = with pkgs; [
264         bashInteractive
265         zsh
266         fish
267       ];
268
269       # scanner support
270       hardware.sane.enable = true;
271
272       hardware.cpu.intel.updateMicrocode = true;
273
274       services.pipewire = {
275         enable = true;
276         alsa.enable = true;
277         alsa.support32Bit = true;
278         pulse.enable = true;
279       };
280
281       # necessary to generate /etc/zsh (so that users can use zsh as a login shell)
282       programs.zsh.enable = true;
283
284       # Define a user account. Don't forget to set a password with ‘passwd’.
285       users.groups.timo = {};
286       users.users.timo = {
287         isNormalUser = true;
288         group = "timo";
289         extraGroups = [
290           "wheel"
291           "networkmanager"
292           "adbusers"
293           "scanner"
294           "docker"
295           "vboxusers"
296           "wireshark"
297           "video" # brightnessctl
298           "libvirtd" # Needed to use virt-manager
```

```
    ];
    uid = 1000;
    shell = "${pkgs.zsh}/bin/zsh";
    # needs to be changed, default is for VMs
    initialPassword = "password";
  };

  systemd.services.channelUpdate  = {
    description = "Updates the unstable channel";
    script = "${pkgs.nix.out}/bin/nix-channel --update";
    startAt = "daily";
    environment.HOME = "/root";
  };


  systemd.services.suspend = {
    description = "Suspend the computer";
    script = ''
      ${pkgs.udev}/bin/systemctl suspend
    '';
  };

  systemd.services.nix-daemon.serviceConfig = {
    MemoryHigh = "6G";
    MemoryMax = "7G";
  };

  system.autoUpgrade = {
    enable = true;
    dates = "daily";
  };

  nix.settings.trusted-users = [ "@wheel" ];
  nix.settings.experimental-features = [ "nix-command" ];
  programs.ssh.knownHosts = {
    aarch64-community-builder = {
      extraHostNames = [ "aarch64.nixos.community" ];
      publicKey = "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIMUTz5i9u5H2FHNAmZJyoJfIGyUm/HfGhfw
    };
  };
  programs.ssh.extraConfig = ''
    Host aarch64-nix-community
      Hostname aarch64.nixos.community
      User timokau
      IdentityFile /root/id_aarch64-builder
  '';

  nix.optimise = {
    automatic = true;
    dates = [ "19:00" ];
  };

  nix.extraOptions = ''
```

```
352         min-free = 2147483648 # automatically collect garbage when <2 GiB free
353         max-free = 3221225472 # stop at 3 GiB
354         max-silent-time = 1800
355         builders-use-substitutes = true
356       '';
357
358       nix.settings.cores = 0; # use all available CPUs
359       nix.settings.max-jobs = 4; # number of jobs (builds) in parallel
360
361       # create a virtual homenet
362       networking.wireguard.interfaces.wg0 = {
363         ips = [ "${wireguard.ip.${config.networking.hostName}}/24" ];
364         listenPort = wireguard.port;
365         privateKeyFile = "/home/timo/wireguard-keys/private"; # FIXME location
366         peers = [
367           {
368             publicKey = wireguard.publicKey.server;
369             allowedIPs = [
370               "${wireguard.ip.server}/32"
371               "${wireguard.ip.pad}/32"
372               "${wireguard.ip.desk}/32"
373               "${wireguard.ip.phone}/32"
374             ];
375             endpoint = "${server_address}:${toString wireguard.port}";
376           }
377         ];
378       };
379     }
```