

sjau /
nixos

<> Code

Issues 1

Pull requests

Actions

Projects

Wiki

Security

nixos / configuration.nix



hyper_ch current system

edfcd33 · 5 years ago



899 lines (785 loc) · 29.7 KB

Code

Blame



Raw



```
1  # Edit this configuration file to define what should be installed on
2  # your system.  Help is available in the configuration.nix(5) man page
3  # and in the NixOS manual (accessible by running 'nixos-help').
4
5  { config, pkgs, ... }:
6
7  let
8
9      # Create file in import path looking like:  { user = 'username'; pass
10     # This info is in a different file, so that the config can be tracked
11     mySecrets = import /root/.nixos/mySecrets.nix;
12
13     #   pass = pkgs: pkgs.pass.override { gnupg = pkgs.gnupg; }; # or your gr
14
15 in
16     # Check if custom vars are set
17
18     # Auth SSH Key
19     assert mySecrets.auth_ssh_key1      != "";
20     assert mySecrets.auth_ssh_key2      != "";
21
22     assert mySecrets.user                != "";
23     assert mySecrets.passwd              != "";
24     assert mySecrets.hashpasswd          != "";
25     assert mySecrets.cifs                != "";
26     assert mySecrets.hostname            != "";
27     assert mySecrets.smbhome              != "";
28     assert mySecrets.smboffice           != "";
29     assert mySecrets.ibsuser              != "";
30     assert mySecrets.ibspass              != "";
31     assert mySecrets.ibsip                != "";
32
33     # Wireguard
34     ## Private Key
35     assert mySecrets.wg_priv_key          != "";
36     ## Home VPN
37     assert mySecrets.wg_home_ip           != "";
```

```

37     assert mySecrets.wg_home_allowed != "";
38     assert mySecrets.wg_home_end != "";
39     assert mySecrets.wg_home_pubkey != "";
40
41     ## Jus-Law VPN
42     assert mySecrets.wg_jl_ips != "";
43     assert mySecrets.wg_jl_allowed != "";
44     assert mySecrets.wg_jl_end != "";
45     assert mySecrets.wg_jl_pubkey != "";
46
47     ## h&b Data VPN
48     assert mySecrets.wg_hb_ips != "";
49     assert mySecrets.wg_hb_allowed != "";
50     assert mySecrets.wg_hb_end != "";
51     assert mySecrets.wg_hb_pubkey != "";
52
53     # ONS
54     assert mySecrets.wg_ons_ips != "";
55     assert mySecrets.wg_ons_allowed != "";
56     assert mySecrets.wg_ons_end != "";
57     assert mySecrets.wg_ons_pubkey != "";
58
59     # SSMTTP
60     assert mySecrets.ssmtp_mailto != "";
61     assert mySecrets.ssmtp_user != "";
62     assert mySecrets.ssmtp_pass != "";
63     assert mySecrets.ssmtp_host != "";
64     assert mySecrets.ssmtp_domain != "";
65     assert mySecrets.ssmtp_root != "";
66
67     {
68         imports =
69             [ # Include the results of the hardware scan.
70               ./hardware-configuration.nix
71               # Fix DisplayLink: see https://gist.github.com/eyJhb/b44a6de73
72               # /root/DisplayLink/displaylink.nix
73             ];
74
75         # Use latest kernel
76         # See VirtualBox settings
77
78         # Add more filesystems
79         boot.supportedFilesystems = [ "zfs" ];
80         boot.zfs.enableUnstable = true;
81         services.zfs.autoScrub = {
82             enable = true;
83             interval = "monthly";
84             pools = [ ]; # List of ZFS pools to periodically scrub. If empty,
85         };
86         services.zfs.zed.settings = {
87             ZED_DEBUG_LOG = "/tmp/zed.dbg.log";
88
89             ZED_EMAIL_ADDR = [ "jaus@sjau.ch" "jau@jus-law.ch" "hyper@servi.hc
90             ZED_EMAIL_PROG = "mail";

```

```

90     ZED_EMAIL_OPTS = "-s '@SUBJECT@' @ADDRESS@";
91
92     ZED_NOTIFY_INTERVAL_SECS = 3600;
93     ZED_NOTIFY_VERBOSE = false;
94
95
96     ZED_USE_ENCLOSURE_LEDS = true;
97     ZED_SCRUB_AFTER_RESILVER = true;
98 };
99
100 # Add memtest86
101 boot.loader.grub.memtest86.enable = true;
102
103 # Use the GRUB 2 boot loader.
104 boot.loader.grub.enable = true;
105 boot.loader.grub.version = 2;
106 # Define on which hard drive you want to install Grub.
107 boot.loader.grub.devices = [
108     "/dev/disk/by-id/ata-Samsung_SSD_850_PRO_1TB_S2BBNWAHC23186P"
109     "/dev/disk/by-id/nvme-Samsung_SSD_970_EVO_Plus_1TB_S4EWNF0M925532F
110 ]; # or "nodev" for efi only
111
112 # Remote ZFS Unlock
113 # boot.initrd.network = {
114 #     enable = false;
115 #     ssh = {
116 #         enable = true;
117 #         port = 2222;
118 #         hostECDSAKey = /root/initrd-ssh-key;
119 #         hostKeys = [
120 #             /etc/secrets/initrd/ssh_host_rsa_key
121 #             /etc/secrets/initrd/ssh_host_ed_25519_key
122 #             "/root/initrd-openssh-key"
123 #         ];
124 #         authorizedKeys = [ "${mySecrets.auth_ssh_key1}" "${mySecrets.
125 #     };
126 #     postCommands = ''
127 #         echo "zfs load-key -a; killall zfs" >> /root.profile
128 #     '';
129 # };
130 boot.initrd.kernelModules = [ "r8169" "cdc_ncm" "xhci_hcd" "usbnet" "i
131 # boot.kernelParams = [ "ip=dhcp" ];
132
133
134 # Clean /tmp at boot
135 boot.cleanTmpDir = true;
136
137
138 # Load additional hardware stuff
139 hardware = {
140     # Hardware settings
141     cpu.intel.updateMicrocode = true;

```

```

142     # enableACPIFirmware = true;
143     enableRedistributableFirmware = true;
144     pulseaudio.enable = true;
145     pulseaudio.package = pkgs.pulseaudioFull;
146     opengl.driSupport32Bit = true; # Required for Steam
147     pulseaudio.support32Bit = true; # Required for Steam
148     bluetooth.enable = true;
149 };
150
151 # Filesystem and remote dirs - thx to sphalerite, clever and Shados
152 fileSystems = let
153   makeServer = { remotefs, userfs, passwordfs, xsystemfs, localfs }: name
154     name = "${localfs}/${name}";
155     value = {
156       device = "://${remotefs}/${name}";
157       fsType = "cifs";
158       options = [ "noauto" "user" "uid=1000" "gid=100" "username=${u
159     };
160   };
161   home = makeServer {
162     remotefs = "${mySecrets.smbhome}";
163     userfs = "${mySecrets.user}";
164     passwordfs = "${mySecrets.cifs}";
165     xsystemfs = "wireguard-wg_home.service";
166     localfs = "/mnt/home";
167   };
168   office = makeServer {
169     remotefs = "${mySecrets.smboffice}";
170     userfs = "none";
171     passwordfs = "none";
172     xsystemfs = "wireguard-wg_jl.service";
173     localfs = "/mnt/jus-law";
174   };
175   in (builtins.listToAttrs (
176     map home [ "Audio" "Shows" "Video" "hyper" "Plex" ]
177     ++ [( office "Advo" )]))
178   // {
179     "/var/tmp" = { device = "tmpfs" ; fsType = "tmpfs"; };
180   };
181
182 # Create some folders
183 system.activationScripts.media = ''
184     mkdir -m 0755 -p /mnt/home/{Audio,Shows,SJ,Video,backup,eeePC,hype
185     mkdir -m 0755 -p /mnt/ibs/{ARCHIV,DATEN,INDIGO,LEAD,VERWALTUNG,SC/
186     mkdir -m 0755 -p /mnt/jus-law/Advo
187 '';
188
189
190
191 # Trust hydra. Needed for one-click installations.
192 nix.trustedBinaryCaches = [ "http://hydra.nixos.org" ];
193
194

```

```

195
196 # Setup networking
197 networking = {
198     # Disable IPv6
199     enableIPv6 = false;
200     hostName = "${mySecrets.hostname}"; # Define your hostname.
201     hostId = "bac8c473";
202     # enable = true; # Enables wireless. Disable when using network
203     networkmanager.enable = true;
204     # interfaces.eth0.useDHCP = true;
205     firewall.enable = true;
206     firewall.allowPing = true;
207     firewall.allowedUDPPorts = [ 137 138 2222 5000 5001 21025 21026 21027 22000 22006 ];
208     firewall.allowedTCPPorts = [ 139 445 2222 3389 5000 5001 21027 22000 22006 ];
209     # Early SSH / initrd
210     # Samba: 137 138 (udp) 139 445 (tcp)
211     # Netcat: 5000
212     # IPerf: 5001
213     # Syncthing: 21025 21026 21027 22000 22006 - WUI: 8384
214     # SPICE/VNC: 5900
215     # WebSockify: 5959
216     # nginx: 4500
217     # RDP: 3389
218     extraHosts = ''
219         188.40.139.2      ns99
220         10.8.0.8          ns
221         176.9.139.175     hetzi manager.roleplayer.org # Hetzner EX4 Ro1
222         10.8.0.97         scriptcase
223         10.8.20.79        raspimam
224         10.8.20.80        mam
225
226         127.0.0.1          subi.home.sjau.ch subi
227         10.10.11.7         vpn-data.jus-law.ch
228         10.10.20.7         wg-data.jus-law.ch
229
230         10.100.200.7       vpn-data.heer-baumgartner.ch
231
232         176.31.121.75      kimsufi ks.jus-law.ch
233         51.15.190.68      ons ons.jus-law.ch
234
235         # Get Ad/Tracking server list from https://github.com/sjau/ads
236         ${builtins.readFile (builtins.fetchurl { name = "blocked_hosts";
237         0.0.0.0             protectedinfoext.biz
238     ''
239 };
240
241 # Enable dbus
242 services.dbus.enable = true;
243
244 # Select internationalisation properties.
245 console.font = "Lat2-Terminus16";
246 console.keyMap = "sg-latin1";

```

```
247     i18n.defaultLocale = "en_US.UTF-8";
248
249     # List services that you want to enable:
250
251
252     # Enable the OpenSSH daemon.
253     services.openssh = {
254         enable = true;
255         permitRootLogin = "yes";
256     };
257
258
259     # Enable CUPS to print documents.
260     services.printing = {
261         enable = true;
262         drivers = [ pkgs.gutenprint pkgs.hplip ];
263     };
264
265
266     # Enable the X11 windowing system.
267     services.xserver = {
268         enable = true;
269         # videoDrivers = [ "amdgpu" "intel" "modesetting" "displaylink" ];
270         videoDrivers = [ "amdgpu" "intel" "modesetting" ];
271         layout = "ch";
272         xkbOptions = "eurosign:e";
273         synaptics = {
274             enable = false;
275         };
276
277         # Enable the KDE Desktop Environment.
278         displayManager.sddm = {
279             enable = true;
280             autoNumlock = true;
281             autoLogin = {
282                 enable = true;
283                 user = "${mySecrets.user}";
284             };
285         };
286         displayManager.lightdm = {
287             enable = false;
288             autoLogin = {
289                 enable = true;
290                 user = "${mySecrets.user}";
291             };
292         };
293         desktopManager.plasma5.enable = true;
294         desktopManager.lxqt.enable = false;
295         windowManager.openbox.enable = false;
296     };
297
298
299     # Use KDE5 unstable
```

```
300     nixpkgs.config.packageOverrides = super: let self = super.pkgs; in {
301         plasma5_stable = self.plasma5_latest;
302         kdeApps_stable = self.kdeApps_latest;
303     };
304
305     # services.xrdp.enable = true;
306     # services.xrdp.defaultWindowManager = "${pkgs.icewm}/bin/icewm";
307     # services.xserver.windowManager.icewm.enable = true;
308
309
310     # Setup a "sendmail"
311     services.ssmtp = {
312         enable = true;
313         authUser = "${mySecrets.ssmtp_user}";
314         authPass = "${mySecrets.ssmtp_pass}";
315         hostName = "${mySecrets.ssmtp_host}";
316         domain = "${mySecrets.ssmtp_domain}";
317         root = "${mySecrets.ssmtp_root}";
318         useSTARTTLS = true;
319         useTLS = true;
320     };
321
322
323     # Enable Virtualbox
324     virtualisation.virtualbox = {
325         host = {
326             enable = true;
327             enableExtensionPack = true;
328         };
329         guest.enable = false;
330     };
331     boot = {
332         # kernelPackages = pkgs.linuxPackages_latest;
333         # Works with EVDI
334         # kernelPackages = pkgs.linuxPackages_5_4;
335         # Testing
336         # kernelPackages = pkgs.linuxPackages_5_5;
337     };
338
339
340     # Enable Docker
341     virtualisation.docker = {
342         enable = true;
343     };
344
345     # Enable Avahi for local domain resoltuion
346     services.avahi = {
347         enable = true;
348         hostName = "${mySecrets.hostname}";
349     };
350
351
```

```

352     # Enable nscd
353     services.nscd = {
354         enable = true;
355     };
356
357     # Enable ntp or rather timesyncd
358     services.timesyncd = {
359         enable = true;
360         servers = [ "0.ch.pool.ntp.org" "1.ch.pool.ntp.org" "2.ch.pool.ntp.org" ];
361     };
362
363     # Custom files in /etc
364     environment.etc = {
365         "easysnap/easysnap.hourly".text = ''
366             # Format: local ds ; local encryption ds ; raw sending ; intermediate ds
367
368             # Servi
369             tankServers/encZFS/home-server/Nixos; tankServers/encZFS;;y;n;r
370             tankServers/encZFS/home-server/Media; tankServers/encZFS;;y;n;r
371             tankMediaBU/encZFS/Plex; tankMediaBU/encZFS;;y;n;r root@servi.hon
372
373             # Remote Servers
374             tankServers/encZFS/online-net-server; tankServers/encZFS;;y;n;r
375             tankServers/encZFS/ovh-cloud-ssd-server; tankServers/encZFS;;y;n;r
376
377             # Roleplayer Server
378             tankServers/encZFS/roleplayer-server/Debian;;;y;n;r root@ispc.rc
379             tankServers/encZFS/roleplayer-server/Debian/home;;;y;n;r root@ispc.rc
380             tankServers/encZFS/roleplayer-server/Debian/var;;;y;n;r root@ispc.rc
381             tankServers/encZFS/roleplayer-server/Debian/var/vmail;;;y;n;r root@ispc.rc
382             tankServers/encZFS/roleplayer-server/Debian/var/www;;;y;n;r root@ispc.rc
383         '';
384         "easysnap/easysnap.hourly".mode = "0644";
385         # Make /etc/hosts writeable
386         "hosts".mode = "0644";
387     };
388
389     # Enable cron
390     services.cron = {
391         enable = true;
392         mailto = "${mySecrets.ssmtp_mailto}";
393         systemCronJobs = [
394             # Make sure network card is set to 1gpbs
395             "* /5 * * * * root ethtool -s enp2s0f1 autoneg on"
396             # Run ZFS Trim every night
397             "1 23 * * * root zpool trim tankSubi"
398             # "0 3,9,15,21 * * * root /root/fstrim.sh >> /tmp/fstrim.txt 2>&
399             "0 2 * * * root /root/backup.sh >> /tmp/backup.txt 2>&
400             "0 */6 * * * root /root/ssd_level_wear.sh >> /tmp/ssd_level_wear.txt 2>&
401             "30 * * * * ${mySecrets.user} pass git pull > /dev/null
402             "40 * * * * ${mySecrets.user} pass git push > /dev/null
403             "* /5 * * * * root autoResilver 'tankSubi' 'usb-Seagate'
404             "* /5 * * * * root autoResilver 'tankSubi' 'usb-Seagate'

```



```

404 #             "25 * * * * * root    autoreilver 'tankSubi' 'usb-TOSHIBA_
405 #             "25 4 * * * root    stopResilver 'tankSubi' 'usb-TOSHIBA_
406 #             "55 * * * * * root    offlineResilver 'tankSubi' 'usb-TOSHIBA_
407             "55 * * * * * root    offlineResilver 'tankSubi' 'usb-Seagate_
408             "3 0 * * * * root    '/root/.acme.sh/acme.sh' --cron --home
409             ### Easy Snap
410             "0 * * * * * root    /home/hyper/Desktop/git-repos/easysnap
411             "25 * * * * * root    /home/hyper/Desktop/git-repos/easysnap
412 ];
413 };
414
415 systemd.services.stopResilver = {
416     description = "Stop Resilvering / Mirroring upon powering down";
417     after = [ "zfs.target" ];
418     wantedBy = [ "zfs.target" ];
419     serviceConfig = {
420         Type = "oneshot";
421         ExecStart = "/run/current-system/sw/bin/true";
422         ExecStop = "/run/current-system/sw/bin/stopResilver 'tankSubi'
423         RemainAfterExit = true;
424     };
425 };
426 systemd.services.stopResilver.enable = true;
427
428
429 #     systemd.services.wireguard-wg_home.serviceConfig.Restart = "on-failure";
430 #     systemd.services.wireguard-wg_home.serviceConfig.RestartSec = "5s";
431 #     systemd.services.wireguard-wg_jl.serviceConfig.Restart = "on-failure";
432 #     systemd.services.wireguard-wg_jl.serviceConfig.RestartSec = "5s";
433 #     systemd.services.wireguard-wg_ons.serviceConfig.Restart = "on-failure";
434 #     systemd.services.wireguard-wg_ons.serviceConfig.RestartSec = "5s";
435
436
437 # Setuid
438 security.wrappers."mount.cifs".source = "${pkgs.cifs-utils}/bin/mount.cifs";
439 security.wrappers."cdrrecord".source = "${pkgs.cdrtools}/bin/cdrrecord";
440 security.wrappers.spice-client-glib-usb-acl-helper.source = "${pkgs.spice-client-glib-usb-acl-helper}/bin/spice-client-glib-usb-acl-helper";
441
442 # Enable sudo
443 security.sudo = {
444     enable = true;
445     wheelNeedsPassword = true;
446 };
447
448
449 # Define a user account. Don't forget to set a password with 'passwd'.
450 users.defaultUserShell = "/var/run/current-system/sw/bin/bash";
451 users.extraUsers.${mySecrets.user} = {
452     isNormalUser = true; # creates home, adds to group users, sets
453     description = "${mySecrets.user}";
454     extraGroups = [ "networkmanager" "vboxusers" "wheel" "audio" "cdrrecord" ];
455     uid = 1000;
456     initialHashedPassword = "${mySecrets.hashedpasswd}";

```

```
457     };
458
459
460     fonts = {
461         enableFontDir = true;
462         enableGhostscriptFonts = true;
463         fonts = with pkgs ; [
464             corefonts
465             liberation_ttf
466             ttf_bitstream_vera
467             dejavu_fonts
468             terminus_font
469             bakoma_ttf
470             clearlyU
471             cm_unicode
472             andagii
473             bakoma_ttf
474             inconsolata
475             gentium
476             ubuntu_font_family
477             source-sans-pro
478             source-code-pro
479         ];
480     };
481
482
483     # Enable Wireguard
484     networking.wireguard.interfaces = {
485         wg_home = {
486             ips = [ "${mySecrets.wg_home_ips}" ];
487             privateKey = "${mySecrets.wg_priv_key}";
488             peers = [ {
489                 allowedIPs = [ "${mySecrets.wg_home_allowed}" ];
490                 endpoint = "${mySecrets.wg_home_end}";
491                 publicKey = "${mySecrets.wg_home_pubkey}";
492                 persistentKeepalive = 25;
493             } ];
494         };
495         wg_ons = {
496             ips = [ "${mySecrets.wg_ons_ips}" ];
497             privateKey = "${mySecrets.wg_priv_key}";
498             peers = [ {
499                 allowedIPs = [ "${mySecrets.wg_ons_allowed}" ];
500                 endpoint = "${mySecrets.wg_ons_end}";
501                 publicKey = "${mySecrets.wg_ons_pubkey}";
502                 persistentKeepalive = 25;
503             } ];
504         };
505         wg_jl = {
506             ips = [ "${mySecrets.wg_jl_ips}" ];
507             privateKey = "${mySecrets.wg_priv_key}";
508             peers = [ {
```

```

509         allowedIPs = [ "${mySecrets.wg_jl_allowed}" ];
510         endpoint = "${mySecrets.wg_jl_end}";
511         publicKey = "${mySecrets.wg_jl_pubkey}";
512         persistentKeepalive = 25;
513     } ];
514 };
515 wg_hb = {
516     ips = [ "${mySecrets.wg_hb_ips}" ];
517     privateKey = "${mySecrets.wg_priv_key}";
518     peers = [ {
519         allowedIPs = [ "${mySecrets.wg_hb_allowed}" ];
520         endpoint = "${mySecrets.wg_hb_end}";
521         publicKey = "${mySecrets.wg_hb_pubkey}";
522         persistentKeepalive = 25;
523     } ];
524 };
525 };
526
527
528 # Enable libvirtd daemon
529 virtualisation.libvirtd = {
530     enable = true;
531     # enableKVM = true;
532     qemuPackage = pkgs.qemu_kvm;
533 };
534 services.spice-vdagentd.enable = true;
535 # Make smartcard reader and label printer accessible to everyone, so t
536 services.udev.extraRules = ''
537     SUBSYSTEM=="usb", ATTR{idVendor}=="072f", ATTR{idProduct}=="90cc",
538     SUBSYSTEM=="usb", ATTR{idVendor}=="04f9", ATTR{idProduct}=="2043",
539     KERNEL=="zd*", SUBSYSTEM=="block", GROUP="users", MODE="0660"
540 '';
541
542
543 # Samba
544 services.samba = {
545     enable = true;
546     securityType = "user";
547     syncPasswordsByPam = true; # Enabling this will add a line direct
548                               # Whenever a password is changed the s
549                               # However, you still have to add the s
550     extraConfig = ''
551         server string = ${mySecrets.hostname}
552         netbios name = ${mySecrets.hostname}
553         workgroup = WORKGROUP
554         max xmit = 65535
555         socket options = TCP_NODELAY IPTOS_LOWDELAY SO_KEEPALIVE
556         hosts allow = 127.0.0. 10.0.0. 10.10.10. 10.10.11. 10.10.20.
557         hosts deny = 0.0.0.0/0
558         security = user
559         guest account = hyper
560         map to guest = bad user
561     # log file = /tmp/%m log

```

```
562     # log level = 3
563
564     # Disable printer
565     printcap name = /dev/null
566     load printers = no
567     printing = bsd
568     show add printer wizard = no
569     disable spoolss = yes
570     '';
571     shares = {
572         Desktop = {
573             path = "/home/hyper/Desktop";
574             browseable = "yes";
575             "read only" = "no";
576             "guest only" = "yes";
577             "guest ok" = "yes";
578             "create mask" = "0644";
579             "directory mask" = "0755";
580             "hosts allow" = "127.0.0.1 10.0.0. 10.10.10. 10.10.11. 10.
581         };
582     };
583 };
584
585
586 # Enable smartmon daemon
587 services.smartd = {
588     enable = true;
589     devices = [ { device = "/dev/sda"; } ];
590 };
591
592
593 # Enable smartcard daemon
594 services.pcscd = {
595     enable = true;
596 };
597
598
599 # Enable Syncthing
600 services.syncthing = {
601     enable = true;
602     dataDir = "/home/${mySecrets.user}/Desktop/Syncthing";
603     configDir = "/home/${mySecrets.user}/.config/syncthing";
604     user = "${mySecrets.user}";
605     openDefaultPorts = true;
606     guiAddress = "0.0.0.0:8384";
607 };
608
609
610 # Enable TOR
611 services.tor = {
612     enable = true;
613     client.enable = true;
```

```

614         controlPort = 9051;
615     };
616
617
618     # Enable sysstat
619     services.sysstat = {
620         enable = true;
621     };
622
623     # Enable Locate
624     services.locate = {
625         enable = true;
626         prunePaths = [ "/tmp" "/var/tmp" "/var/cache" "/var/lock" "/var/run" ];
627     };
628
629
630     # Time.
631     time.timeZone = "Europe/Zurich";
632
633
634     # Add the NixOS Manual on virtual console 8
635     services.nixosManual.showManual = true;
636
637
638     # Enable ALSA
639     sound.enable = true;
640
641     # Setup bash completion
642     programs.bash.enableCompletion = true;
643
644
645     # Setup nano
646     programs.nano.nanorc = ''
647         set nowrap
648         set tabstospaces
649         set tabsize 4
650         set constantshow
651         # include /usr/share/nano/sh.nanorc
652     '';
653
654     # Setup ADB
655     programs.adb.enable = true;
656     nixpkgs.config.android_sdk.accept_license = true;
657
658     # The NixOS release to be compatible with for stateful data such as databases
659     # It will e.g. upgrade databases to newer versions and that can't be rolled back
660     system.stateVersion = "19.03";
661
662     nixpkgs.config.allowUnfree = true;
663     # nixpkgs.config.allowBroken = true;
664
665     nixpkgs.config.chromium = {
666         enable = true;
667         # Chromium removed support for WebGL 1
668         # enableWebGL1 = false;
669     };

```

```
666 # enable pepper flash = true; # chromium removed support for mozilla
667 };
668
669
670 # List packages installed in system profile. To search by name, run:
671 # $ nix-env -qaP | grep wget
672 # List of packages that gets installed....
673 environment.systemPackages = with pkgs; [
674 #     androidenv.platformTools # contains ADB
675 #     android-studio
676     aspell
677     aspellDicts.de
678     aspellDicts.en
679     audacity
680     bash-completion
681     bind          # provides dig and nslookup
682     bluedevil
683     bluez
684     bluez-tools
685     brave
686     cargo
687     chromium
688     cifs_utils
689     cdrtools
690     cmake
691     coreutils
692     cryptsetup
693     curl
694     dcfldd # dd alternative that shows progress and can make different
695     dialog
696 #     displaylink
697     directvnc
698     dmidecode
699     dos2unix
700     dstat
701     easysnap
702     enca
703     ethtool
704     exfat
705     fatrace
706     file
707     filezilla
708     firefoxWrapper
709     ffmpeg
710     foo2zjs          # Printer drivers for Oki -> http://foo2hi
711     foomatic-filters
712     gcc
713     gdb
714     ghostscript
715     gimp
716     git
717     gksu
718     gnome3.dconf
```

```
719         gnome3.dconf-editor
720         gnome3.zenity
721         gnupg             # GnuPG 2 -> provides gpg2 binary
722         gparted
723         gptfdisk
724         gwenview
725         hdparm
726         htop
727         hunspellDicts.de-ch
728         icedtea8_web
729         iftop
730         imagemagick
731         iosevka
732         iotop
733         iperf
734         iputils
735         jdk
736         jpegoptim
737         jq
738         jre
739     # KDE 5
740         ark
741         dolphin
742         kdenlive     frei0r # frei0r provides transition effects
743         kdeFrameworks.kdesu
744         k3b
745         kate
746         kcalc
747         konversation
748         okular
749         opusTools
750         oxygen
751         oxygen-icons5
752         oxygenfonts
753         plasma-desktop
754         plasma-integration
755         plasma-nm
756         plasma-workspace
757         spectacle # KSnapshot replacement for KDE 5
758         kdeApplications.kdialog
759         kdeApplications.krfb
760     # End of KDE 5
761         kvm
762         libreoffice
763         libuchardet
764         lightning
765         links
766         lshw
767         lsof
768         lxqt.lximage-qt
769         manpages
770         mc
```

```
771     mdadm
772     mediainfo
773     mkpasswd
774     mktorrent
775     mplayer
776     mpv
777     ms-sys
778     # netcat-gnu
779     ninja
780     nix-index
781     nix-info
782     # nix-index # provides nix-locate
783     nix-prefetch-github
784     # nix-repl # do: :l <nixpkgs> to load the packages, then do qt5.m
785     nmap
786     nox      # Easy search for packages
787     nss
788     nssTools
789     ntfs3g
790     nvme-cli
791     opensc
792     openssl
793     openvpn
794     pandoc
795     parted
796     pass
797     patchelf
798     pavucontrol
799     pciutils
800     pcsctools
801     pdftk
802     php      # PHP-Cli
803     pinentry
804     pinentry-qt
805     pkgconfig
806     # playonlinux
807     poppler_utils # provides command_not_found
808     pv
809     python27Packages.youtube-dl
810     python37Packages.websockify
811     psmisc
812     pwgen
813     qemu
814     qt5Full
815     qtpass
816     recode
817     recoll
818     rfkill
819     rustc
820     simplescreenrecorder
821     smartmontools
822     smem
823     smplayer
```



```

824     sox
825     spice
826     spice-gtk
827     win-spice
828     sqlite
829     sqlitebrowser
830     sshpass
831     stdenv # build-essential on nixos
832     #     steam
833     subversion
834     sudo
835     #     suisseid-pkcs11
836     swt
837     sylpheed
838     syncthing
839     sysfsutils
840     sysstat
841     system_config_printer
842     teamspeak_client
843     telnet
844     tesseract
845     #     (import (builtins.fetchTarball ("https://github.com/NixOS/nixpkgs
846     thunderbird
847     birdtray
848     tightvnc
849     tmux
850     unoconv
851     unrar
852     unzip
853     usbutils
854     virt-viewer
855     virtmanager
856     vlc
857     wget
858     which
859     whois
860     wine
861     winetricks
862     wireguard
863     wireshark
864     woeusb
865     #     xpdf # provides pdftotext
866     zip
867
868     # easysnap
869     (pkgs.callPackage /home/hyper/Desktop/git-repos/nix-expressions/ea
870
871     (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
872     (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
873     (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
874     #     (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
875     (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com

```

```
876         (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
877         (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
878         (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
879         (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
880         (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
881         (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
882
883     #         (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
884     #         (pkgs.callPackage (builtins.fetchurl "https://raw.githubusercontent.com
885
886     #         (pkgs.callPackage ./localsigner.nix {})
887     #         (pkgs.callPackage ./suisseid-pkcs11.nix {})
888     #         (pkgs.callPackage ./swisssign-pin-entry.nix {})
889     #         (pkgs.callPackage ./swisssigner.nix {})
890     #     ] ++ ( builtins.filter pkgs.stdenv.lib.isDerivation (builtins.attrValues
891
892     ];
893
894     # suisseid-pkcs11 requires on ubuntu the following packages:
895     # fontconfig fontconfig-config fonts-dejavu-core libaudio2 libccid libfont
896     # libqt4-network libqt4-script libqt4-sql libqt4-xml libqt4-xmlpatterns lib
897     # libxt6 pcscd qtcore4-l10n suisseid-pkcs11 swisssign-pin-entry x11-common
898
899 }
```