ERIC BAILEY

DOTFILES

Contents

Flake

The main entrypoint flake.nix[1] defines a Nix Flake[2]. At the top-level, the flake defines `inputs` and `outputs`.

[1] https://github.com/yurrriq/dotfiles/blob/main/flake.nix
[2] https://nixos.wiki/wiki/Flakes

⟨*flake.nix*⟩≡
```
{

  description = "My (semi-)literate, Nix-based dotfiles";

  inputs = {
    ⟨inputs⟩
  };

  outputs = { self, ... }@inputs:
    let
      inherit (inputs.nixpkgs) lib;
      ⟨outputs variables⟩
    in
    {
      ⟨system-agnostic outputs⟩
    } // inputs.flake-utils.lib.eachDefaultSystem (system:
      let
        ⟨per-system outputs variables⟩
      in
      {
        ⟨per-system outputs⟩
      });
}
```

Inputs

emacs-overlay

The Emacs overlay[3] "comes with extra functions to generate an Emacs closure from various types of dependency declaration." Notably, `emacsWithPackagesFromUsePackage`, which generates an Emacs closure from an Emacs config file, by way of `use-package`[4]. See the Emacs section on page 49 for usage. Follow the top-level `flake-utils` and `nixpkgs` to dedupe `inputs`.

[3] https://github.com/nix-community/emacs-overlay

[4] https://jwiegley.github.io/use-package/

⟨*inputs*⟩≡
```
emacs-overlay = {
  url = "github:nix-community/emacs-overlay";
  inputs.flake-utils.follows = "flake-utils";
  inputs.nixpkgs.follows = "nixpkgs";
};
```

flake-utils

flake-utils[5] is a set of pure Nix functions that are useful when writing flakes. Pin a version to dedupe `inputs`.

[5] https://github.com/numtide/flake-utils

⟨*inputs*⟩+≡
```
flake-utils.url = "github:numtide/flake-utils";
```

home-manager

home-manager[6] is an invaluable tool for managing a deterministic `$HOME`. Pin a version and follow the top-level `nixpkgs` and `flake-utils` to dedupe `inputs`.

[6] https://github.com/nix-community/home-manager

⟨*inputs*⟩+≡
```
home-manager = {
  url = "github:nix-community/home-manager/release-22.11";
  inputs.nixpkgs.follows = "nixpkgs";
  inputs.utils.follows = "flake-utils";
};
```

nixGL

nixGL[7] is a wrapper tool that solves the OpenGL problem with Nix.[8] Pin a version and follow the top-level `flake-utils` and `nixpkgs` to dedupe `inputs`. See `nixGLWrap` for usage.

[7] https://github.com/guibou/nixGL

[8] https://github.com/NixOS/nixpkgs/issues/9415

⟨*inputs*⟩+≡
```
nixgl = {
  url = "github:guibou/nixGL";
  inputs.flake-utils.follows = "flake-utils";
  inputs.nixpkgs.follows = "nixpkgs";
};
```

nixos-hardware

nixos-hardware[9] is "a collection of NixOS modules covering hardware quirks."

[9] https://github.com/NixOS/nixos-hardware

⟨*inputs*⟩+≡
```
nixos-hardware.url = "github:nixos/nixos-hardware";
```

nixpkgs

Someday soon again, prefer to live on the edge and pin `nixpkgs` to the *nixpkgs-unstable* branch.

⟨*inputs*⟩+≡
```
nixpkgs.url = "github:nixos/nixpkgs/release-22.11";
```

⟨*inputs*⟩+≡
```
# nixpkgs-stable.url = "github:nixos/nixpkgs/release-22.11";
```

NUR

The Nix User Repository[10] provides "user contributed Nix packages." Notably, I use some of Robert Helgesson's Firefox add-ons[11] (see Firefox on page 55).

[10] https://github.com/nix-community/NUR

[11] https://gitlab.com/rycee/nur-expressions/-/blob/master/pkgs/firefox-addons/default.nix

⟨*inputs*⟩+≡
```
nur.url = "github:nix-community/nur";
```

treefmt

⟨*inputs*⟩+≡
```
treefmt-nix = {
  inputs.nixpkgs.follows = "nixpkgs";
  url = "github:numtide/treefmt-nix";
};
```

Outputs

Overlays

Use `composeManyExtensions`[12] to compose all the other ⟨*overlays*⟩ into one, `overlays.default`.

[12] "Compose several extending functions of the type expected by `extends` into one where changes made in preceding functions are made available to subsequent ones." — NixOS/nixpkgs@c3b35f21

⟨*system-agnostic outputs*⟩≡
```
overlays = {
  default =
    lib.composeManyExtensions
      (lib.attrValues
        (lib.filterAttrs (name: _: name ≠ "default") self.overlays));
  ⟨overlays⟩
};
```

Per-system pkgs

For each `system`, define a package collection that composes the overlays from `deadnix` and `emacs-overlay`, as well as the locally-defined `home-manager` and `noweb` overlays.

⟨*per-system outputs variables*⟩≡
```
pkgs = import inputs.nixpkgs {
  overlays = [
    inputs.emacs-overlay.overlay
    self.overlays.home-manager
    self.overlays.noweb
  ];
  inherit system;
};
```

Development environments

Define some development environments. One of the main benefits is the super fast direnv[13] integration.

[13] https://direnv.net

⟨*per-system outputs*⟩≡
```
devShells = {
  ⟨devShells⟩
};
```

Define the default `devShell`, including tools to build the PDF,[14] manage secrets[15], generate passwords[16], bump the version[17], pre-commit[18]), and manage the non-Nix symlink farm[19].

[14] https://yurrriq.codes/dotfiles/dotfiles.pdf
[15] https://github.com/mozilla/sops
[16] https://linux.die.net/man/1/mkpasswd
[17] https://github.com/fsaintjacques/semver-tool
[18] https://pre-commit.com
[19] https://www.gnu.org/software/stow/

⟨*devShells*⟩≡
```
default = pkgs.mkShell {
  inherit (self.packages.${system}.default) FONTCONFIG_FILE;
  buildInputs = with pkgs; [
    biber
    (
      emacsWithPackagesFromUsePackage {
        alwaysEnsure = true;
        config = ./config/emacs/init.el;
      }
    )
    editorconfig-checker
    git
    git-lfs
    gnumake
    gnupg
    home-manager
    mkpasswd
    nodePackages.node2nix
    pre-commit
    semver-tool
    sops
    stow
  ] ++ self.packages.${system}.default.nativeBuildInputs;
};
```

Configure a development environment for my XMonad[20] config. Use the Emacs overlay to configure a nice Emacs setup using lsp-haskell[21], Ormolu[22] to format Haskell code, and include pointfree[23] for… reasons.

[20] https://xmonad.org
[21] https://emacs-lsp.github.io/lsp-haskell/
[22] https://github.com/tweag/ormolu
[23] https://github.com/bmillwood/pointfree

⟨*devShells*⟩+≡
```
xmonad =
  let
    myXMonad =
      pkgs.haskellPackages.callCabal2nix "my-xmonad" ./config/xmonad { };
  in
  pkgs.mkShell {
    buildInputs = with pkgs; [
      cabal-install
      (
        emacsWithPackagesFromUsePackage {
          alwaysEnsure = true;
          config = ./config/xmonad/emacs.el;
        }
      )
      ghcid
      haskell-language-server
      haskellPackages.ormolu
      haskellPackages.pointfree
      pre-commit
    ] ++ myXMonad.env.nativeBuildInputs;
  };
```

Formatters

Use treefmt[24] (via treefmt-nix[25]) to format code, as a pre-commit hook or by calling `nix fmt` directly.

[24] https://github.com/numtide/treefmt
[25] https://github.com/numtide/treefmt-nix

⟨*per-system outputs*⟩+≡
```
formatter = inputs.treefmt-nix.lib.mkWrapper pkgs {
  projectRootFile = "flake.nix";
  programs = {
    ⟨deadnix⟩
    ⟨nixpkgs-fmt⟩
    ⟨shellcheck⟩
    ⟨shfmt⟩
  };
  settings.formatter = rec {
    ⟨deadnix settings⟩
    ⟨nixpkgs-fmt settings⟩
    ⟨shellcheck settings⟩
    ⟨shfmt settings⟩
  };
};
```

Use deadnix[26] to scan Nix files for dead code (unused variable bindings).

[26] https://github.com/astro/deadnix

⟨*deadnix*⟩≡
```
deadnix = {
  enable = true;
  no-lambda-arg = true;
  no-lambda-pattern-names = true;
};
```

Use the same `excludes` from the ⟨*nixpkgs-fmt settings*⟩, to exclude some generated Nix files.

⟨*deadnix settings*⟩≡
```
deadnix.excludes = nixpkgs-fmt.excludes;
```

Use nixpkgs-fmt[27] to format Nix files.

[27] https://github.com/nix-community/nixpkgs-fmt

⟨*nixpkgs-fmt*⟩≡
```
nixpkgs-fmt.enable = true;
```

⟨*nixpkgs-fmt settings*⟩≡
```
nixpkgs-fmt.excludes = [
  "machines/*/hardware-configuration.nix"
  "pkgs/development/node-packages/node-env.nix"
  "pkgs/development/node-packages/node-packages.nix"
];
```

Use ShellCheck[28] to lint shell scripts.

[28] https://www.shellcheck.net/

⟨*shellcheck*⟩≡
```
shellcheck.enable = true;
```

⟨*shellcheck settings*⟩≡
```
shellcheck = {
  includes = [
    "*.sh"
    ".envrc"
  ];
  options = [
    "-format=tty"
    "-shell=bash"
  ];
};
```

Use shfmt[29] to format shell scripts.

[29] https://github.com/mvdan/sh

⟨*shfmt*⟩≡
```
shfmt.enable = true;
```

Configure my preferred shfmt settings.

⟨*shfmt settings*⟩≡
```
shfmt = {
  includes = [
    "*.sh"
    ".envrc"
  ];
  options = [
    "-i"
    "4"
    "-ci"
    "-s"
    "-w"
  ];
};
```

Packages

The default package builds the PDF[30]. See also: Packages.

[30] https://yurrriq.codes/dotfiles/dotfiles.pdf

⟨*per-system outputs*⟩+≡
```
packages = {
  default = self.packages.${system}.yurrriq-dotfiles;
  yurrriq-dotfiles = pkgs.callPackage ./. { };
};
```

Modules

See Modules.

⟨*system-agnostic outputs*⟩+≡
```
nixosModules = {
  ⟨Modules⟩
};
```

NixOS configurations

Define system configurations for my personal laptop and my work laptop.

⟨*system-agnostic outputs*⟩+≡
```
nixosConfigurations = {
  "nixps" = mkSystem "nixps" "dell-xps-15-9560-intel";
  "MSP-EBAILEY01" = mkSystem "sruxps" "dell-xps-13-7390";
};
```

home-manager modules

⟨*system-agnostic outputs*⟩+≡
```
homeManagerModules =
  let
    inherit (builtins) attrNames baseNameOf filter listToAttrs pathExists readDir toPath;
    inherit (lib.attrsets) nameValuePair;
    inherit (lib.strings) hasSuffix removeSuffix;
    resolveModule = relativePath:
      nameValuePair
        (removeSuffix ".nix" (baseNameOf relativePath))
        (import (toPath (./config + ("/" + relativePath))));
    isModule = path:
      hasSuffix ".nix" path ||
      pathExists (./config + ("/" + path + "/default.nix"));
  in
  listToAttrs
    (map resolveModule
      (filter isModule
        (attrNames (readDir ./config))));
```

home-manager configurations

Describe this

⟨*system-agnostic outputs*⟩+≡

```
homeConfigurations.eric = inputs.home-manager.lib.homeManagerConfiguration {
  modules = self.nixosModules.home-manager.home-manager.sharedModules ++ [
    self.nixosModules.nixRegistry
    ./machines/sruxps/home.nix
    {
      home = {
        username = "eric";
        homeDirectory = "/home/eric";
        stateVersion = "22.11";
      };
    }
  ];
  pkgs = import inputs.nixpkgs {
    inherit (self.nixosModules.nixpkgs.nixpkgs) config;
    overlays = self.nixosModules.nixpkgs.nixpkgs.overlays ++ [
      # https://github.com/nix-community/home-manager/issues/2251#issuecomment-895338427
      (final: prev: {
        kitty = prev.lib.nixGLWrap { pkg = prev.kitty; };
        # FIXME
        # zoom-us = prev.lib.nixGLWrap { pkg = prev.zoom-us; binName = "zoom"; };
      })
    ];
    system = "x86_64-linux";
  };
};
```

Variables

Define a helper function `mkSystem :: String → String → AttrSet` for creating system configurations. Given a (NixOS system) `name` and a `machine` name, which corresponds to directories in machines/[31] containing `configuration.nix` and `hardware-configuration.nix`.

[31] https://github.com/yurrriq/dotfiles/tree/main/machines

⟨*outputs variables*⟩≡

```
mkSystem = name: machine: lib.nixosSystem {
  modules = [
    (./machines + "/${name}/hardware-configuration.nix")
    inputs.nixos-hardware.nixosModules.${machine}
    inputs.nixos-hardware.nixosModules.common-pc-laptop-ssd
    inputs.home-manager.nixosModules.home-manager
    self.nixosModules.home-manager
    self.nixosModules.location
    self.nixosModules.nix
    self.nixosModules.nixPath
    self.nixosModules.nixRegistry
    self.nixosModules.nixos
    self.nixosModules.nixpkgs
    self.nixosModules.bootyjams
    self.nixosModules.virtualisation
    (./machines + "/${name}/configuration.nix")
  ];
  system = "x86_64-linux";
};
```

`pkgNameElem` takes a list of package names and returns a function suitable for an `allowUnfreePredicate`[32].

[32] https://nixos.org/manual/nixpkgs/unstable/#sec-allow-unfree

```
mkSystem :: [String] -> (String -> Bool)
```

⟨*outputs variables*⟩+≡
```
pkgNameElem = names: pkg:
  builtins.elem (lib.getName pkg) names;
```

Overlays

Define an overlay with a pinned version of home-manager.

⟨*overlays*⟩≡
```
home-manager = final: prev: {
  home-manager = inputs.home-manager.packages.${prev.system}.home-manager;
};
```

OpenGL is a kind of a nightmare on non-NixOS, and nixGL makes it Just Work™. Define a convenience function `nixGLWrap :: AttrSet -> Derivation` to wrap the wrapper. Since most of my machines have Intel video cards and I haven't adopted Nvidia drivers, use `nixGLIntel`.

⟨*overlays*⟩+≡
```
nixGLWrap = final: prev: {
  lib = prev.lib // {
    nixGLWrap = { pkg, binName ? prev.lib.getName pkg }:
      prev.writeShellScriptBin binName ''
        exec ${final.nixgl.nixGLIntel}/bin/nixGLIntel ${pkg}/bin/${binName} "$@"
      '';
  };
};
```

Define an overlay that adds my custom `nodePackages`.

```
nodePackages = final: prev: {
  nodePackages =
    unstable-pkgs.nodePackages // prev.callPackage ./pkgs/development/node-packages {
      inherit (prev) pkgs nodejs;
    };
};
```

Define an overlay that overrides noweb[33] to build with a non-graphical version of Icon[34].

[33] https://www.cs.tufts.edu/~nr/noweb/
[34] https://www2.cs.arizona.edu/icon/

⟨*overlays*⟩+≡
```
noweb = final: prev: {
  noweb = prev.noweb.override {
    icon-lang = prev.icon-lang.override {
      withGraphics = false;
    };
  };
};
```

NixOS Modules

Document (or better yet: refactor) the other `nixosModules`

⟨*Modules*⟩≡
```
bootyjams = import ./modules/bootyjams.nix;
```

⟨*Modules*⟩+≡
```
home-manager = {
  home-manager = {
    sharedModules =
      let
        excluded = [
          "bugwarrior"
          "nix"
          "screen-locker"
          "taskwarrior"
        ];
        notExcluded = lib.filterAttrs (name: _: !(builtins.elem name excluded));
      in
      builtins.attrValues (notExcluded self.homeManagerModules);
    useGlobalPkgs = true;
    useUserPackages = true;
    verbose = true;
  };
};
```

⟨*Modules*⟩+≡
```
location = import ./modules/location.nix;
nix = import ./modules/nix.nix;
```

Document this `nixPath` hack

⟨*Modules*⟩+≡
```
nixPath = {
  nix.nixPath = lib.mapAttrsToList (n: v: "${n}=${v}")
    (lib.filterAttrs (n: _: n ≠ "self") inputs) ++ [
    "nixos-config=/etc/nixos/configuration.nix"
  ];
};
```

Document `nixRegistry`

⟨*Modules*⟩+≡
```
nixRegistry = {
  nix.registry = lib.mapAttrs (_: flake: { inherit flake; }) inputs;
};
```

⟨*Modules*⟩+≡
```
nixos = import ./modules/nixos.nix;
```

Document `nixpkgs` config

⟨*Modules*⟩+≡

```
nixpkgs = {
  nixpkgs.config.allowUnfreePredicate = pkgNameElem [
    "Oracle_VM_VirtualBox_Extension_Pack"
    "lastpass-password-manager"
    "nvidia"
    "reaper"
    "slack"
    "spotify"
    "spotify-unwrapped"
    "steam"
    "steam-original"
    "steam-run"
    "zoom"
  ];
  nixpkgs.overlays = [
    self.overlays.default
    inputs.emacs-overlay.overlay
    inputs.nixgl.overlay
    inputs.nur.overlay
  ];
};
```

See Virtualisation.

⟨*Modules*⟩+≡

```
virtualisation = import ./modules/virtualisation.nix;
```

Packages

The yurrriq-dotfiles (PDF) derivation

⟨*default.nix*⟩≡

```
{ stdenv
, nix-gitignore
, makeFontsConf
, iosevka
, gawk
, noweb
, python3Packages
, texlive
, which
}:

stdenv.mkDerivation rec {
  pname = "yurrriq-dotfiles";
  version = builtins.readFile ./VERSION;
  src = nix-gitignore.gitignoreSource [
    ".git/"
    "docs"
    "result*"
    "machines/*/secrets/"
  ] ./.;

  FONTCONFIG_FILE = makeFontsConf {
    fontDirectories = [ iosevka ];
  };

  configurePhase = "
    substituteInPlace ./bin/fix-underscores \
        -replace '/usr/bin/env -S gawk' '${gawk}/bin/gawk'
  ";

  nativeBuildInputs = [
    gawk
    noweb
    python3Packages.pygments
    (
      texlive.combine {
        inherit noweb;
        inherit (texlive) scheme-small
          braket
          catchfile
          datatool
          datetime
```

```
            dirtytalk
            fancyref
            fmtcount
            framed
            frankenstein
            fvextra
            glossaries
            glossaries-extra
            hardwrap
            ifplatform
            latexmk
            mathpazo
            mfirstuc
            minted
            substr
            titlesec
            tkz-base
            todonotes
            tufte-latex
            xetex
            xindy
            xfor
            xstring
            ;
        }
      )
      which
    ];

    makeFlags = [
      "PREFIX=${placeholder "out"}"
    ];

  }
```

Additional Node packages

Augment the upstream `nodePackages`. At the moment, add only codeowners[35].

[35] https://www.npmjs.com/package/codeowners

⟨*pkgs/development/node-packages/node-packages.json*⟩≡

```
  [
    "codeowners"
  ]
```

Machines

NiXPS

NixOS configuration

The NiXPS configuration is an expression that takes, at least, `config`, `lib`, and `pkgs`.

⟨*machines/nixps/configuration.nix*⟩≡
```
{ lib, pkgs, ... }:
let
  username = "yurrriq";
in
{
  ⟨Set the location to MSP⟩
  ⟨Blacklist a few kernel modules⟩
  ⟨Configure the LUKS device⟩
  ⟨Configure the environment⟩
  ⟨Configure the file systems⟩
  ⟨Tweak hardware config⟩
  ⟨Configure Bluetooth⟩
  ⟨Configure PulseAudio and OpenGL⟩
  ⟨Configure home-manager⟩
  ⟨Configure networking⟩
  ⟨Configure Nix⟩
  ⟨Configure sudo⟩
  ⟨Configure X Server⟩
  ⟨Configure user⟩
  environment.systemPackages = with pkgs; [
    xorg.xbacklight
  ];
}
```

Configure the `location` *module*.

⟨*Set the location to MSP*⟩≡
```
airportCode = "MSP";
```

⟨*Blacklist a few kernel modules*⟩≡
```
boot.blacklistedKernelModules = [ "nouveau" "nvidia" "psmouse" ];
```

Configure the *LUKS*[36] devices.

[36] https://gitlab.com/cryptsetup/cryptsetup/

⟨*Configure the LUKS device*⟩≡
```
boot.initrd.luks.devices.root.device = "/dev/nvme0n1p2";
```

Include `~/bin/` in `$PATH`.

⟨*Configure the environment*⟩≡
```
environment.homeBinInPath = true;
```

Link some extra paths.

⟨*Configure the environment*⟩+≡
```
environment.pathsToLink = [
  "/lib/aspell"
  "/share/emacs/site-lisp"
  "/share/fish"
];
```

⟨*Configure the file systems*⟩≡
```
fileSystems."/" = {
  device = "/dev/disk/by-uuid/024a1168-9949-4cb2-bbd1-4b19a9d49ef2";
  fsType = "ext4";
};

fileSystems."/boot" = {
  device = "/dev/disk/by-uuid/7574-B246";
  fsType = "vfat";
};

fileSystems."/var/lib/docker/plugins" = {
  device = "/var/lib/docker/plugins";
  fsType = "none";
  options = [ "bind" ];
};

fileSystems."/var/lib/docker/overlay2" = {
  device = "/var/lib/docker/overlay2";
  fsType = "none";
  options = [ "bind" ];
};
```

⟨*Tweak hardware config*⟩≡
```
hardware.bumblebee.enable = false;
hardware.nvidiaOptimus.disable = false;
```

⟨*Configure Bluetooth*⟩≡
```
services.blueman.enable = true;
hardware.bluetooth = {
  enable = true;
  settings = {
    General = {
      Enable = "Source,Sink,Media,Socket";
    };
  };
};
```

Tweak PulseAudio and OpenGL settings for Steam[37].

[37] https://store.steampowered.com/linux

⟨*Configure PulseAudio and OpenGL*⟩≡
```
hardware.pulseaudio.support32Bit = true;
hardware.opengl.driSupport32Bit = true;
```

⟨*Configure home-manager*⟩≡
```
home-manager.users."${username}" = import ./home.nix;
```

⟨*Configure networking*⟩≡
```
networking.firewall = {
  enable = true;
  allowedTCPPortRanges = [
    # { from = 8000; to = 8000; }
  ];
};

networking.hostName = "nixps";
```

⟨*Configure Nix*⟩≡
```
nix = {
  settings = {
    cores = 8;
    trusted-users = [ "root" username ];
  };
};
```

⟨*Configure sudo*⟩≡
```
security.sudo = {
  enable = true;
  extraConfig = "
    ${username} ALL=(ALL) NOPASSWD: ALL
  ";
};
```

⟨*Configure X Server*⟩≡
```
services.xserver = {
  displayManager.autoLogin = {
    enable = true;
    user = username;
  };
  monitorSection = "
    DisplaySize 508 285
  ";
  dpi = 220;
};
```

⟨*Configure user*⟩≡
```
users.mutableUsers = false;
users.users."${username}" = {
  name = username;
  hashedPassword = lib.fileContents "/etc/nixos/secrets/${username}.hashedPassword";
  isNormalUser = true;
  extraGroups = [
    "audio"
    "disk"
    "docker"
    "http"
    "networkmanager"
    "systemd-journal"
    "video"
    "wheel"
  ];
  uid = 1000;
  shell = "/etc/profiles/per-user/${username}/bin/fish";
};
```

home-manager configuration

⟨*machines/nixps/home.nix*⟩≡
```
{ pkgs, ... }:
{
  ⟨Configure email accounts⟩
  ⟨Add some more user packages⟩
  home.stateVersion = "22.11";
  services.picom.enable = true;
}
```

⟨*Configure email accounts*⟩≡
```
accounts.email.accounts = {
  personal = {
    address = "eric@ericb.me";
    gpg.key = "F88372B24A806FF23BCB3A4E2DDDF8606958B3F9";
    primary = true;
    realName = "Eric Bailey";
  };
  work.address = "e.bailey@sportradar.com";
};
```

⟨*Add some more user packages*⟩≡
```
home.packages = with pkgs; [
  amdvlk
  appimage-run
  calibre
  fd
  frescobaldi
  gnutls
  lutris
  musescore
  openscad
  powertop
  protontricks
  reaper
  signal-desktop
  steam
  tellico
  winetricks
  zoom-us
];
```

SRUXPS

NixOS configuration

The SRUXPS configuration is an expression that takes, at least, `config`, `lib`, and `pkgs`.

⟨*machines/sruxps/configuration.nix*⟩≡
```
{ config, lib, pkgs, ... }:
let
  username = "e.bailey";
in
{
  ⟨Set the location⟩
  ⟨Configure the LUKS devices⟩
  ⟨Configure vm.swappiness⟩
  ⟨Add some kernel modules⟩
  boot.kernelParams = [ "acpi_rev_override" "mem_sleep_default=deep" "intel_iommu=igfx_off" ];
  ⟨Link some extra paths⟩
  ⟨Install system-wide packages⟩
  ⟨Configure the file systems⟩
  ⟨Configure Bluetooth⟩
  hardware.acpilight.enable = true;
  hardware.opengl.enable = true;
  ⟨Configure home-manager⟩
  ⟨Configure networking⟩
  ⟨Configure Nix⟩
  ⟨Configure sudo⟩
  ⟨Configure services⟩
  ⟨Configure user⟩
  virtualisation.virtualbox.host.enable = true;
}
```

Describe the location module

⟨*Set the location*⟩≡
```
airportCode = "MSP";
```

Configure the *LUKS* devices.

⟨*Configure the LUKS devices*⟩≡
```
boot.initrd.luks.devices = {
  cryptkey.device = "/dev/disk/by-uuid/2a44a760-206c-448d-a126-527b8b63f5d0";

  cryptroot = {
    device = "/dev/disk/by-uuid/6cd51820-547b-4378-b566-47f8cdbc46df";
    keyFile = "/dev/mapper/cryptkey";
  };

  cryptswap = {
    device = "/dev/disk/by-uuid/7d80e701-3a6b-4bb0-b8a3-dd5dfb432cdd";
    keyFile = "/dev/mapper/cryptkey";
  };
};
```

https://docs.oracle.com/en/operating-systems/oracle-linux/7/admin/section_h4v_3yt_tr.html

⟨*Configure vm.swappiness*⟩≡
```
boot.kernel.sysctl = {
  "vm.swappiness" = 1;
};
```

⟨*Add some kernel modules*⟩≡
```
boot.kernelModules = [
  "coretemp"
  "i915.enable_psr=0"
];
```

⟨*Link some extra paths*⟩≡
```
environment.homeBinInPath = true;
environment.pathsToLink = [
  "/lib/aspell"
  "/share/emacs/site-lisp"
  "/share/fish"
  # FIXME: "/share/icons"
];
```

⟨*Install system-wide packages*⟩≡
```
environment.systemPackages = with pkgs; [
  kubelogin
];
```

btrfs clean boot?

⟨*Configure the file systems*⟩≡
```
fileSystems."/" = {
  device = "/dev/disk/by-uuid/75e8b8ef-8143-4f93-a60b-c5d53adb80d3";
  fsType = "ext4";
  options = [ "noatime" "nodiratime" "discard" ];
};

fileSystems."/boot" = {
  device = "/dev/disk/by-uuid/DF46-7ADE";
  fsType = "vfat";
};
```

⟨*Configure Bluetooth*⟩≡
```
hardware.bluetooth = {
  enable = true;
  settings = {
    General = {
      Enable = "Source,Sink,Media,Socket";
    };
  };
};
```

⟨*Configure home-manager*⟩≡
```
home-manager.users."${username}" = import ./home.nix;
```

⟨*Configure networking*⟩≡
```
networking.hostName = "MSP-EBAILEY01";
```

⟨*Configure Nix*⟩≡
```
nix = {
  binaryCaches = [
    "https://sportradar.cachix.org"
  ];
  binaryCachePublicKeys = [
    "sportradar.cachix.org-1:6MyCzOfUMeMTxU5QnogkyYOBtr5f5atW/qeS+TjmUfE="
  ];
  trustedUsers = [ "root" username ];
};
```

⟨*Configure sudo*⟩≡
```
security.sudo = {
  enable = true;
  extraConfig = ”
    ${username} ALL=(ALL) NOPASSWD: ALL
  ”;
};
```

⟨*Configure services*⟩≡
```
services.blueman.enable = true;

services.fwupd.enable = true;

services.lorri.enable = false;

services.thermald.enable = lib.mkForce false; # FIXME

services.tlp.enable = true;

services.xserver = {
  config = ”
    Section ”Device”
      Identifier  ”Intel Graphics”
      Driver      ”intel”
      Option      ”TearFree”        ”true”
      Option      ”SwapbuffersWait” ”true”
      BusID       ”PCI:0:2:0”
    EndSection
  ”;

  displayManager.autoLogin = {
    enable = true;
    user = username;
  };

  monitorSection = ”
    DisplaySize 406 228
  ”;

  screenSection = ”
    Option         ”AllowIndirectGLXProtocol” ”off”
    Option         ”TripleBuffer” ”on”
  ”;
};
```

⟨*Configure user*⟩≡
```
users.mutableUsers = false;
users.users."${username}" = {
  name = username;
  hashedPassword = lib.fileContents "/etc/nixos/secrets/${username}.hashedPassword";
  isNormalUser = true;
  extraGroups = [
    "audio"
    "disk"
    "docker"
    "http"
    "networkmanager"
    "systemd-journal"
    "vboxusers"
    "video"
    "wheel"
  ];
  uid = 1001;
  shell = "/etc/profiles/per-user/${username}/bin/fish";
};
```

home-manager configuration

⟨*machines/sruxps/home.nix*⟩≡
```
{ config, lib, pkgs, ... }:
{
  ⟨Import configs⟩
  ⟨Configure email accounts⟩
  ⟨Configure Docker credentials⟩
  ⟨Configure a convenient high-level .envrc⟩
  ⟨Configure the keyboard⟩
  home.packages = with pkgs; [
    ⟨Install some user packages⟩
  ];
  ⟨Configure picom⟩
  ⟨Miscellaneous config⟩
}
```

⟨*Import configs*⟩≡
```
imports = [
  ../../config/nix.nix
  ../../config/screen-locker.nix
];
```

⟨*Configure email accounts*⟩≡
```
accounts.email.accounts = {
  personal.address = "eric@ericb.me";
  work = {
    address = "e.bailey@sportradar.com";
    gpg.key = "86BAD22D1F8DBBEC486C49012C32D5C1C17A8045";
    primary = true;
    realName = "Eric Bailey";
  };
};
```

⟨*Configure Docker credentials*⟩≡
```
home.file.".docker/config.json".text = ''
  {
      "credHelpers": {
          "docker.io": "pass",
          "gitlab.sportradar.ag:4567": "pass"
      }
  }
'';
```

⟨*Configure a convenient high-level .envrc*⟩≡
```
home.file."src/gitlab.sportradar.ag/.envrc".text = ''
  case $(kubectl config current-context) in
      *k8s.srus*|sapi*nov1*)
          export AWS_PROFILE=msp
          ;;
      *)
          export AWS_PROFILE=default
  esac

  export CI_SERVER_HOST=gitlab.sportradar.ag
  export CI_JOB_TOKEN=$(pass "$CI_SERVER_HOST"/token/api)
  export CI_REGISTRY="$CI_SERVER_HOST":4567
  export CI_REGISTRY_USER=gitlab-ci-token
  export CI_REGISTRY_PASSWORD="$CI_JOB_TOKEN"
  export GITLAB_TOKEN="$CI_JOB_TOKEN"
  export JIRA_API_TOKEN="$(pass jira.sportradar.ag/e.bailey)"
  export JIRA_AUTH_TYPE=bearer
  export GITLAB_REGISTRY_ACCESS_TOKEN="$CI_REGISTRY_PASSWORD"
'';
```

⟨*Configure the keyboard*⟩≡
```
home.keyboard = {
  options = [
    "ctrl:nocaps"
    "compose:ralt"
  ];
};
```

Install some tools for authenticating with AWS and Azure.

- The AWS IAM Authenticator is used for interacting with EKS clusters.
- EC2 Instance Connect allows you to use your AWS CLI credentials to SSH into EC2 instances.
- AWS Azure Login ...
- naal[38] is my tool for non-interactive AWS Azure login.

[38] https://github.com/yurrriq/naal

⟨*Install some user packages*⟩≡
```
aws-iam-authenticator
awscli2
python3Packages.ec2instanceconnectcli
```

Install some miscellaneous tools.

- ...
- powertop might improve battery life.
- I haven't used Dhall in a while, but it's still interesting.
- iw[39] is useful for troubleshooting wireless issues, like in Trondheim.

[39] https://wireless.wiki.kernel.org/en/users/documentation/iw

⟨*Install some user packages*⟩+≡
```
btop
fd
gomplate
powertop
progress
scc
```

⟨*⟩≡
```
dhall
dhall-json
iw
```

Install some Docker tools.

- `docker-credential-pass` for store Docker credentials with `pass`.
- I might need Docker Compose[40] again someday, but not lately. Plus, it doesn't (currently) work with Podman.

[40] https://docs.docker.com/compose/

⟨*Install some user packages*⟩+≡
```
docker-credential-helpers
```

⟨*⟩+≡
```
docker-compose
```

⟨*Install some user packages*⟩+≡
```
# TODO: podman-compose
```

Install some convenient Kubernetes tools.

⟨*Install some user packages*⟩+≡
```
krew
kubectl
kubectx
kubelogin
kustomize
lens
stern
```

⟨*Install some user packages*⟩+≡
```
vault
```

Install some tools for testing alert pipelines.

⟨*Install some user packages*⟩+≡
```
fortune
prometheus-alertmanager
```

Install some miscellaneous GUI programs.

- Zoom is annoying but useful.
- It would be cool to play with Nyxt[41] some lazy day.

[41] https://nyxt.atlas.engineer/

⟨*⟩+≡
```
# FIXME
# zoom-us
```

⟨*⟩+≡
```
nyxt
```

Super Productivity[42] is a nice tool for time/work tracking and syncing with Jira.

[42] https://super-productivity.com

⟨*⟩+≡
```
super-productivity
```

Organize this better

⟨*Install some user packages*⟩+≡
```
bind
curl
httpie
cachix
nixUnstable
home-manager
nixgl.nixGLIntel
networkmanager
```

⟨*Configure picom*⟩≡
```
services.picom = {
  enable = true;
  settings = {
    unredir-if-possible = true;
  };
  vSync = true;
};
```

⟨*Miscellaneous config*⟩≡
```
home.sessionPath = [
  "${config.home.homeDirectory}/bin"
];

nix.enable = true;

programs.kitty.settings.font_size = 12;

services.random-background = {
  enable = true;
  imageDirectory = "/usr/share/backgrounds/";
  display = "scale";
};

targets.genericLinux.enable = true;

xresources.properties = {
  "Xft.dpi" = 290;
};
```

Modules

Location

⟨*modules/location.nix*⟩≡
```
{ config, lib, ... }:
let
  airportCode = config.airportCode;
in
{
  ⟨Airport codes⟩
  config = lib.mkMerge [
    ⟨Configure the location⟩
  ];
}
```

Configure the list of airport codes for places I tend to work. The default is `"MSP"`, since that's where I live.

⟨*Airport codes*⟩≡
```
options.airportCode = lib.mkOption {
  default = "MSP";
  type = lib.types.enum [ "ATL" "LHR" "LJU" "MSP" "TRD" ];
};
```

When I'm in Minneapolis I usually[43] work from home, but sometimes I work from the Sportradar office in Minneapolis.

[43] always during a pandemic

⟨*Configure the location*⟩≡
```
(
  lib.mkIf (airportCode == "MSP") {
    location = {
      latitude = 44.98;
      longitude = -93.27;
    };
    time.timeZone = "America/Chicago";
  }
)
```

Obviously I don't actually work from the Cook Out on Ponce de Leon Ave in Atlanta, but I do sometimes work from Decatur, the ATL airport, or Boone, and this is close enough for time zones.

⟨*Configure the location*⟩+≡
```
(
  lib.mkIf (airportCode == "ATL") {
    location = {
      latitude = 33.77;
      longitude = -84.37;
    };
    time.timeZone = "America/New_York";
  }
)
```

Sometimes I visit the Sportradar office in London, UK.

⟨*Configure the location*⟩+≡
```
(
  lib.mkIf (airportCode == "LHR") {
    location = {
      latitude = 51.51;
      longitude = -0.09;
    };
    time.timeZone = "Europe/London";
  }
)
```

Sometimes I visit the Sportradar office in Ljubljana, Slovenia.

⟨*Configure the location*⟩+≡
```
(
  lib.mkIf (airportCode == "LJU") {
    location = {
      latitude = 46.09;
      longitude = 14.55;
    };
    time.timeZone = "Europe/Ljubljana";
  }
)
```

Sometimes I visit the Sportradar office in Trondheim, Norway.

⟨*Configure the location*⟩+≡
```
(
  lib.mkIf (airportCode == "TRD") {
    location = {
      latitude = 63.43;
      longitude = 10.40;
    };
    time.timeZone = "Europe/Oslo";
  }
)
```

NixOS

This modules manages some common configuration on NixOS systems.

⟨*modules/nixos.nix*⟩≡
```
{ lib, pkgs, ... }:

{
  ⟨Configure boot loader⟩
  ⟨Configure the console font and keyboard settings⟩
  ⟨Configure location⟩
  ⟨Enable networkmanager⟩
  ⟨Configure services⟩
  ⟨Configure audio⟩
  system.stateVersion = "22.11";
}
```

Delete all files in `/tmp` during boot, and steal some useful bits from `xps-common.nix`.

⟨*Configure boot loader*⟩≡
```
boot = {
  cleanTmpDir = true;
  loader.systemd-boot.enable = lib.mkDefault true;
  loader.efi.canTouchEfiVariables = lib.mkDefault true;
};
```

Configure the console to use a more legible font, and to use the keymap from the X server keyboard settings. Notably this means caps lock will behave as I expect (remapped to control).

⟨*Configure the console font and keyboard settings*⟩≡
```
console.font = "Lat2-Terminus16";
console.useXkbConfig = true;
```

Link/document this

⟨*Configure location*⟩≡
```
location.provider = "manual";
```

Use NetworkManager (intead of WPA Supplicant).

⟨*Enable networkmanager*⟩≡
```
networking.networkmanager.enable = true;
```

Hibernate when the laptop lid is closed, mount the Keybase filesystem, enable Redshift, and configure the X server.

⟨*Configure services*⟩≡
```
services = {
  logind.lidSwitch = "hibernate";
  kbfs.enable = true;
  redshift = {
    enable = true;
    temperature.night = 2300;
  };
  thermald.enable = lib.mkDefault true;
  ⟨Configure the X server⟩
};
```

Enable the X server, start it automatically,

⟨*Configure the X server*⟩≡
```
xserver = {
  enable = true;

  autorun = true;

  ⟨Configure the display manager⟩

  ⟨Configure libinput⟩

  videoDrivers = lib.mkDefault [ "intel" ];

  xkbOptions = "ctrl:nocaps,compose:ralt";
};
```

Enable lightdm as the display manager.

⟨*Configure the display manager*⟩≡
```
displayManager = {
  lightdm.enable = true;
  session = [
    ⟨Configure xmonad as the window manager⟩
  ];
};
```

Use the workaround described on Discourse to run xmonad from home-manager.

⟨*Configure xmonad as the window manager*⟩≡
```
{
  name = "home-manager";
  manage = "window";
  start = "
    ${pkgs.runtimeShell} $HOME/.hm-xsession &
    waitPID=$!
  ";
}
```

⟨*Configure libinput*⟩≡
```
libinput = {
  enable = true;
  touchpad = {
    accelSpeed = "1.0";
    disableWhileTyping = true;
    naturalScrolling = false;
    tapping = true;
  };
};
```

Enable the PulseAudio soudn server, including extra Bluetooth modules. Also enable ALSA sound.

⟨*Configure audio*⟩≡
```
hardware.pulseaudio = {
  enable = true;
  # FIXME
  # extraModules = [ pkgs.pulseaudio-modules-bt ];
  package = pkgs.pulseaudioFull;
};
sound.enable = true;
```

Nix

⟨*modules/nix.nix*⟩≡
```
{ pkgs, ... }:
{
  nix = {
    settings = {
      ⟨Configure binary caches⟩
    };
    # TODO: daemonNiceLevel = 19;
    ⟨Configure extra options⟩
    ⟨Configure garbage collection⟩
    optimise.automatic = true;
    package = pkgs.nixUnstable;
  };
  ⟨Install Cachix⟩
}
```

Install Cachix[44] and configure my personal binary cache[45].

[44] https://cachix.org
[45] https://yurrriq.cachix.org

⟨*Install Cachix*⟩≡
```
environment.systemPackages = with pkgs; [
  cachix
];
```

⟨*Configure binary caches*⟩≡
```
substituters = [
  "https://yurrriq.cachix.org"
];
trusted-public-keys = [
  "yurrriq.cachix.org-1:evpJ5wKluf7QOCcv69VkIxCOtHgubrqXlZpp3JAXLBE="
];
```

Enable Flakes and the Nix command.

⟨*Configure extra options*⟩≡
```
extraOptions = ''
  experimental-features = flakes nix-command
'';
```

Configure garbage collection to run automatically and delete generations older than 30 days.

⟨*Configure garbage collection*⟩≡
```
gc = {
  automatic = true;
  options = "--delete-older-than 30d";
};
```

Virtualisation

⟨*modules/virtualisation.nix*⟩≡
```
{ config, lib, pkgs, ... }:
{
  ⟨Install crun⟩
  virtualisation = {
    ⟨Configure Docker⟩
    ⟨Configure Podman⟩
    ⟨Configure VirtualBox⟩
  };
}
```

Document podman vs dockerd

Docker

Prefer to disable the Docker daemon.

⟨*Configure Docker*⟩≡
```
docker = {
  enable = lib.mkDefault false;
  liveRestore = lib.mkDefault false;
};
```

Podman

It seems there is a bug in podman such that it doesn't properly handle `~/.config/containers/containers.conf`, so install `crun` system-wide.

```
WARN[0000] Found default OCIruntime /nix/store/.../bin/crun path which is missing from [engine.runtimes] in container
```

⟨*Install crun*⟩≡
```
environment.systemPackages = lib.optionals config.virtualisation.podman.enable [ pkgs.crun ];
```

Install Podman by default, if the Docker daemon is disabled.

⟨*Configure Podman*⟩≡
```
podman = {
  enable = lib.mkDefault (!config.virtualisation.docker.enable);
  dockerCompat = lib.mkDefault true;
};
```

VirtualBox

⟨*Configure VirtualBox*⟩≡
```
virtualbox.host.enable = lib.mkDefault false;
virtualbox.host.enableExtensionPack = lib.mkDefault (config.virtualisation.virtualbox.host.enable);
```

⟨*modules/bootyjams.nix*⟩≡ `% ⟹ this file was generated automatically by noweave — better not edit it`

```
{ config, lib, pkgs, ... }:
let
  username = lib.head (lib.attrNames config.home-manager.users);
in
{
  environment.systemPackages = with pkgs; [
    nfs-utils
  ];
  fileSystems =
    let
      mkMount = path: {
        device = "//192.168.1.147${path}";
        fsType = "cifs";
        options = [
          "credentials=/etc/nixos/secrets/bootyjams.club"
          "gid=${toString config.ids.gids.users}"
          "noauto"
          "rw"
          "uid=${toString config.users.users.${username}.uid}"
          "vers=2.0"
          # "x-systemd.device-timeout=5s"
          "x-systemd.idle-timeout=600"
          "x-systemd.mount-timeout=5s"
        ];
      };
    in
    {
      "/mnt/music" = mkMount "/homes/eric/music";
      "/mnt/reaper" = mkMount "/REAPER Media";
    };
}
```

Software configuration

Applications

⟨*config/applications.nix*⟩≡
```
{ lib, pkgs, ... }:
{
  home.packages = with pkgs; (
    [
      ⟨Common applications⟩
    ]
  ) ++ lib.optionals stdenv.isLinux (
    [
      ⟨Linux applications⟩
    ]
  );
}
```

Common applications

Clementine[46] is my favorite music player for local (network) files, and Spotify is my streaming service of choice.

[46] https://www.clementine-player.org

⟨*Common applications*⟩≡
```
clementine
spotify
```

I find Pulse Audio Volume Control[47] to be a necessary evil, for the moment.

[47] https://gitlab.freedesktop.org/pulseaudio/pavucontrol

⟨*Common applications*⟩+≡
```
pavucontrol
```

Slack is great for communicating with open source communities, and we also use it at work, so it's convenient to just install it everywhere.

⟨*Common applications*⟩+≡
```
slack
```

Linux-specific applications

Install qpdfview[48] only on Linux.

[48] https://launchpad.net/qpdfview

⟨*Linux applications*⟩≡
```
qpdfview
```

Bash

Enable the Bash module.

⟨*config/bash.nix*⟩≡
```
{ ... }:

{

  programs.bash.enable = true;

}
```

bat

Enable the bat[49] module.

[49] https://github.com/sharkdp/bat

⟨*config/bat.nix*⟩≡
```
{ ... }:

{

  programs.bat = {
    enable = true;
    config = {
      ⟨Configure bat⟩
    };
  };

}
```

Use YAML syntax for `.yaml.gotmpl` files.

⟨*Configure bat*⟩≡
```
map-syntax = "*.yaml.gotmpl:YAML";
```

For the pager, use `less`.

⟨*Configure bat*⟩+≡
```
pager = "less -FR";
```

Set the style to `changes`.

⟨*Configure bat*⟩+≡
```
style = "changes";
```

Use the Monokai Extended theme.

⟨*Configure bat*⟩+≡
```
theme = "Monokai Extended";
```

Browserpass

⟨*config/browserpass.nix*⟩≡
```
{ pkgs, ... }:

{

  home.packages = with pkgs; [
    browserpass
  ];

  programs.browserpass = {
    enable = true;
    browsers = [ "firefox" ];
  };

}
```

bugwarrior

⟨*config/bugwarrior.nix*⟩≡
```
{ config, pkgs, ... }:

{
```

⟨*config/bugwarrior.nix*⟩+≡
```
  home.packages = with pkgs; [
    bugwarrior
  ];
```

⟨*config/bugwarrior.nix*⟩+≡
```
  imports = [
    ./password-store.nix
    ./taskwarrior
  ];
```

⟨*config/bugwarrior.nix*⟩+≡

```
    programs.taskwarrior = {
      config = {
        context.other = "jiraurl.none or -work";
        context.work = "jiraurl.any or +work";
        uda = {
          jiracreatedts = {
            label = "Created At";
            type = "date";
          };
          jiradescription = {
            label = "Jira Description";
            type = "string";
          };
          jiraestimate = {
            label = "Estimate";
            type = "numeric";
          };
          jirafixversion = {
            label = "Fix Version";
            type = "string";
          };
          jiraid = {
            label = "Jira Issue ID";
            type = "string";
          };
          jiraissuetype = {
            label = "Issue Type";
            type = "string";
          };
          jirastatus = {
            label = "Jira Status";
            type = "string";
          };
          jirasummary = {
            label = "Jira Summary";
            type = "string";
          };
          jiraurl = {
            label = "Jira URL";
            type = "string";
          };
        };
      };
    };
```

⟨*config/bugwarrior.nix*⟩+≡

```
    xdg.configFile."bugwarrior/bugwarriorrc".text = ''
      [general]
      targets = sportradar_jira
      taskrc = ${config.home.homeDirectory}/.taskrc
      inline_links = False
      annotation_links = True
      annotation_comments = True
      legacy_matching = False
      log.level = DEBUG
      log.file = ${config.home.homeDirectory}/log/bugwarrior.log
      annotation_length = 80

      [sportradar_jira]
      service = jira
      jira.base_uri = https://jira.sportradar.ag
      jira.username = e.bailey
      jira.password = @oracle:eval:pass jira.sportradar.ag
      jira.query = ((assignee = currentUser() OR reporter = currentUser()) OR (summary ~ currentUser() OR description
      jira.version = 8
      jira.add_tags = work
      jira.description_template = {{jiraid}}: {{jirasummary}}
    '';
```

⟨*config/bugwarrior.nix*⟩+≡

```
  }
```

direnv

Configure direnv[50], including bash and fish integration, by default, and disable the zsh integration, since I don't use zsh.

[50] https://direnv.net

⟨*config/direnv.nix*⟩≡

```
  { ... }:

  {

    programs.direnv = {
      enable = true;
      enableZshIntegration = false;
    };

  }
```

Dunst

⟨*config/dunst.nix*⟩≡
```
{ pkgs, ... }:

{
  services.dunst = {
    enable = true;
    ⟨Configure Dunst icon theme⟩
    settings = {
      global = {
        ⟨Configure Dunst global settings⟩
      };
      ⟨Configure Dunst shortcuts⟩
      ⟨Configure Dunst urgency styling⟩
      ⟨Configure critical Slack notifications⟩
    };
  };
}
```

⟨*Configure Dunst icon theme*⟩≡
```
iconTheme = {
  package = pkgs.paper-icon-theme;
  name = "Paper";
  size = "48x48";
};
```

⟨*Configure Dunst global settings*⟩≡
```
font = "Iosevka Term 10";
markup = "yes";
plain_text = "no";
format = "<b>%s</b>\\n%b";
sort = "yes";
indicate_hidden = "yes";
alignment = "center";
bounce_freq = 0;
show_age_threshold = 30;
word_wrap = "yes";
ignore_newline = "no";
stack_duplicates = "yes";
hide_duplicates_count = "yes";
origin = "bottom-right";
width = "300";
height = "100";
notification_limit = 3;
offset = "15x70";
shrink = "no";
transparency = 15;
idle_threshold = 0;
follow = "keyboard";
sticky_history = "yes";
history_length = 15;
show_indicators = "no";
startup_notification = false;
dmenu = "/run/current-system/sw/bin/dmenu -p dunst:";
browser = "/etc/profiles/per-user/e.bailey/bin/firefox -new-tab";
icon_position = "left";
max_icon_size = 80;
frame_width = 0;
frame_color = "#8EC07C";
```

⟨*Configure Dunst shortcuts*⟩≡
```
shortcuts = {
  close = "mod4+space";
  close_all = "mod4+mod1+space";
  # context = "ctrl+shift+period";
  history = "ctrl+grave";
};
```

⟨*Configure Dunst urgency styling*⟩≡
```
urgency_low = {
  frame_color = "#3B7C87";
  foreground = "#3B7C87";
  background = "#2B313C";
  timeout = 4;
};
urgency_normal = {
  frame_color = "#5B8234";
  foreground = "#5B8234";
  background = "#2B313C";
  timeout = 6;
};
urgency_critical = {
  frame_color = "#B7472A";
  foreground = "#B7472A";
  background = "#191311";
  timeout = 8;
};
```

⟨*Configure critical Slack notifications*⟩≡
```
slack = {
  appname = "Slack";
  body = "*critical*";
  frame_color = "#B7472A";
  foreground = "#B7472A";
  background = "#191311";
  urgency = "critical";
};
```

Emacs

⟨*config/emacs/default.nix*⟩≡
```
{ config, lib, pkgs, ... }:

{

  home.file.".emacs.d/init.el".source = ./init.el;

  home.packages = with pkgs; [
    graphviz
    noweb
    sqlite
  ];

  home.sessionVariables = rec {
    EDITOR = "emacsclient -nw -a \"\"";
    GIT_EDITOR = EDITOR;
    VISUAL = "emacsclient -cna \"\"";
  };

  programs.emacs = {
    enable = true;
    package = pkgs.emacsWithPackagesFromUsePackage {
      alwaysEnsure = true;
      config = ./init.el;
      override = epkgs: epkgs // {
        noweb-mode = pkgs.noweb;
      };
    };
  };

  programs.fish.shellAliases = lib.mkIf (config.programs.fish.enable) rec {
    e = "emacsclient -na \"\"";
    ec = e + " -c";
    et = "emacsclient -nw -a \"\"";
  };

  services.emacs.enable = ! pkgs.stdenv.isDarwin;

}
```

Init File

⟨*config/emacs/init.el*⟩≡
```
(server-start)
(setq server-window 'pop-to-buffer-same-window)
```

⟨*config/emacs/init.el*⟩+≡
```
(column-number-mode 1)
```

⟨*config/emacs/init.el*⟩+≡
```
(add-to-list 'exec-path "/run/current-system/sw/bin")
```

⟨*config/emacs/init.el*⟩+≡
```
(menu-bar-mode 0)
(scroll-bar-mode 0)
(tool-bar-mode 0)
```

⟨*config/emacs/init.el*⟩+≡
```
(set-face-attribute 'default t :family "Iosevka" :height 100)
```

⟨*config/emacs/init.el*⟩+≡
```
(require 'package)
```

⟨*config/emacs/init.el*⟩+≡
```
(setq-default frames-only-mode t
              indent-tabs-mode nil
              package-archives nil
              package-enable-at-startup nil)
```

⟨*config/emacs/init.el*⟩+≡
```
(package-initialize)

(eval-when-compile
  (require 'use-package))

(setq-default use-package-always-defer t
              use-package-always-ensure t)
```

https://stackoverflow.com/a/18330742

⟨*config/emacs/init.el*⟩+≡
```
(let ((backup-directory (concat user-emacs-directory "backup")))
  (unless (file-exists-p backup-directory)
    (make-directory backup-directory t))
  (setq backup-directory-alist `(("" . ,backup-directory))))

(setq auto-save-default         t
      auto-save-interval        200
      auto-save-timeout         20
      backup-by-copying         t
      delete-by-moving-to-trash t
      delete-old-versions       t
      kept-new-versions         6
      kept-old-versions         2
      make-backup-files         t
      vc-make-backup-files      t
      version-control           t)
```

https://anirudhsasikumar.net/blog/2005.01.21.html Copyright ©2004-2011 Anirudh Sasikumar. All rights reserved.

⟨*config/emacs/init.el*⟩+≡

```
(define-minor-mode sensitive-mode
  "For sensitive files like password lists.
It disables backup creation and auto saving.

With no argument, this command toggles the mode.
Non-null prefix argument turns on the mode.
Null prefix argument turns off the mode."
  :init-value nil
  :lighter " "
  (if (symbol-value sensitive-mode)
      (progn
        ;; disable backups
        (set (make-local-variable 'backup-inhibited) t)
        ;; disable auto-save
        (if auto-save-default
            (auto-save-mode -1)))
    ;; resort to default value of backup-inhibited
    (kill-local-variable 'backup-inhibited)
    ;; resort to default auto save setting
    (if auto-save-default
        (auto-save-mode 1))))
```

⟨*config/emacs/init.el*⟩+≡

```
(dolist (pattern '("^\\(/dev/shm/\\|/tmp/\\)"
                   "\\.\\(enc\\|gpg\\|hashedPassword\\)$"))
  (add-to-list 'auto-mode-alist (cons pattern 'sensitive-mode)))
```

⟨*config/emacs/init.el*⟩+≡

```
(setq custom-file "~/.emacs.d/private/local/custom.el")
```

⟨*config/emacs/init.el*⟩+≡

```
(load-theme 'wombat)
```

⟨*config/emacs/init.el*⟩+≡

```
(global-set-key (kbd "C-x C-k") 'kill-this-buffer)
(global-set-key (kbd "s-u") 'revert-buffer)
```

⟨*config/emacs/init.el*⟩+≡
```
  (use-package better-defaults)

  (use-package direnv
    :ensure t)

  (use-package dockerfile-mode)

  (use-package fill-column-indicator
    :config
    (setq-default fill-column 80)
    (global-display-fill-column-indicator-mode))

  ;; (use-package frames-only-mode)

  (use-package kubernetes-tramp)

  (use-package paredit)

  (use-package rainbow-delimiters)
  (use-package rainbow-mode)
```

⟨*config/emacs/init.el*⟩+≡
```
  (use-package elixir-mode)
  (use-package fish-mode)
  (use-package gap-mode)
  (use-package go-mode)
  (use-package haskell-mode)
  (use-package idris-mode)
  (use-package j-mode)
  (use-package markdown-mode)
  (use-package nix-mode
    :mode ("\\.nix\\'"))
  (use-package rust-mode)
  (use-package terraform-mode)
```

⟨*config/emacs/init.el*⟩+≡
```
  (setq c-default-style      "k&r"
        c-basic-offset       4
        emacs-lisp-mode-hook '(fci-mode
                               paredit-mode
                               rainbow-delimiters-mode)
        js-indent-level      2
        text-mode-hook       '(text-mode-hook-identify))
```

⟨*config/emacs/init.el*⟩+≡
```
  (require 'org-tempo)

  (org-babel-do-load-languages
   'org-babel-load-languages
   '((shell . t)))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package avy
  :demand
  :config
  (global-set-key (kbd "C-;") 'avy-goto-char)
  (global-set-key (kbd "C-'") 'avy-goto-char-2)
  (global-set-key (kbd "M-g f") 'avy-goto-line))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package crux
  :demand
  :config (global-set-key (kbd "C-a") 'crux-move-beginning-of-line))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package deadgrep
  :demand
  :config (global-set-key (kbd "M-s-f") #'deadgrep))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package direnv)
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package editorconfig
  :ensure t
  :config
  (editorconfig-mode 1))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package emojify)
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package graphviz-dot-mode
  :config
  (setq graphviz-dot-indent-width 4))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package hl-todo
  :demand
  :config (global-hl-todo-mode t))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package magit
  :demand
  ;; FIXME: :config (global-magit-file-mode t)
  )
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package multiple-cursors
  :demand
  :config (global-set-key (kbd "C-S-c C-S-c") 'mc/edit-lines))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package noweb-mode
  :load-path "/run/current-system/sw/share/emacs/site-lisp"
  :mode ("\\.nw\\'")
  :demand)
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package nyan-mode
  :demand
  :config (nyan-mode 1))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package smex
  :demand
  :config
  (global-set-key (kbd "M-x") 'smex)
  (global-set-key (kbd "M-X") 'smex-major-mode-commands)
  (global-set-key (kbd "C-c C-c M-x") 'execute-extended-command))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package tuareg
  :mode ("\\.ml\\'" "\\.mli\\'"))
```

⟨*config/emacs/init.el*⟩+≡
```
(use-package whitespace-cleanup-mode
  :demand
  :config (global-whitespace-cleanup-mode t))
```

Use `yaml-mode`, even for `.yaml.gotmpl` files.

⟨*config/emacs/init.el*⟩+≡
```
(use-package yaml-mode
  :config
  (add-to-list 'auto-mode-alist '("\\.yaml.gotmpl\\'" . yaml-mode)))
```

Firefox

⟨*config/firefox.nix*⟩≡

```
{ pkgs, ... }:

{

  programs.firefox = {
    enable = true;
    extensions = with pkgs.nur.repos.rycee.firefox-addons; [
      browserpass
      darkreader
      lastpass-password-manager
      privacy-badger
    ];
    profiles = {
      default = {
        settings = {
          # http://kb.mozillazine.org/About:config_entries
          "browser.ctrlTab.recentlyUsedOrder" = false;
          "browser.newtabpage.activity-stream.asrouter.userprefs.cfr.addons" = false;
          "browser.newtabpage.activity-stream.asrouter.userprefs.cfr.features" = false;
          "browser.newtabpage.activity-stream.feeds.section.topstories" = false;
          "browser.newtabpage.activity-stream.feeds.snippets" = false;
          # FIXME: "browser.newtabpage.activity-stream.improvesearch.topSiteSearchShortcuts.havePinned" = null;
          "browser.newtabpage.activity-stream.section.highlights.includePocket" = false;
          "browser.newtabpage.activity-stream.showSearch" = false;
          # FIXME: "browser.newtabpage.pinned" = "[]";
          # FIXME: "browser.search.defaultenginename" = "DuckDuckGo";
          # FIXME: "browser.search.defaulturl" = "https://duckduckgo.com/?q=";
          "browser.search.hiddenOneOffs" = "Google,Bing,Amazon.com,eBay,Twitter,Wikipedia (en)";
          # FIXME: git "browser.search.selectedEngine" = "DuckDuckGo";
          "browser.search.suggest.enabled" = false;
          "browser.startup.page" = 3;
          "browser.tabs.unloadOnLowMemory" = true;
          "browser.urlbar.placeholderName" = "DuckDuckGo";
          "extensions.activeThemeID" = "firefox-compact-dark@mozilla.org";
          "font.size.variable.x-western" = 12;
          "signon.rememberSignons" = false;
        };
      };
    };
  };

}
```

fish

⟨*config/fish/default.nix*⟩≡ `% ⟹ this file was generated automatically by noweave — better not edit it`

```
{ lib, pkgs, ... }:

{

  imports = [
    ./abbrs.nix
    ./aliases.nix
  ];

  home = {
    packages = with pkgs; [
      exa
    ];
    sessionVariables = {
      SHELL = "fish";
      TERMINAL = "kitty";
    };
  };

  programs.fish =
    let
      inherit (lib.strings) fileContents;
    in
    {
      enable = true;
      interactiveShellInit = fileContents ./interactiveShellInit.fish;
      shellInit = fileContents ./shellInit.fish;
    };

  programs.zoxide.enable = true;

}
```

Fish abbreviations

"`abbr` manages abbreviations - user-defined words that are replaced with longer phrases after they are entered.

For example, a frequently-run command like `git checkout` can be abbreviated to `gco`. After entering `gco` and pressing Space or Enter, the full text `git checkout` will appear in the command line."

⟨*config/fish/abbrs.nix*⟩≡

```
{ ... }:

{

  programs.fish.shellAbbrs = {
    ⟨Direnv fish abbreviations⟩
    ⟨Kubernetes fish abbreviations⟩
    ⟨Nix fish abbreviations⟩
    ⟨ripgrep fish abbreviations⟩
  };

}
```

DIRENV[51] IS GREAT for directory-specific environments, and saving keystrokes is great, too.

[51] https://direnv.net/

⟨*Direnv fish abbreviations*⟩≡
```
da = "direnv allow";
dn = "direnv deny";
dr = "direnv reload";
```

`kubectl` IS VERY CUMBERSOME to type all the time. Using fish abbrevations, save some precious keystrokes. It's definitely worth the time.

⟨*Kubernetes fish abbreviations*⟩≡
```
kc = "kubectl";
kcd = "kubectl drain -delete-emptydir-data -ignore-daemonsets";
kcn = "kubectl -namespace";
kcnp = "kubectl get pods -field-selector=spec.nodeName=";
kcx = "kubectl -context";
kg = "kubectl get";
kgp = "kubectl get pods";
kgy = "kubectl get -o yaml";
kn = "kubens";
kns = "kubens";
krr = "kubectl rollout restart";
krs = "kubectl rollout status";
kt = "stern";
kx = "kubectx";
```

IT'S USEFUL TO RUN `nix build` with various flags. Why not save some keystrokes for those tasks, too?

⟨*Nix fish abbreviations*⟩≡
```
nb = "nix build";
nbd = "nix build -dry-run";
nbn = "nix build -no-link";
nbo = "nix build -o";
```

I'M NOT QUITE USED TO all the `ripgrep` flags yet. I'm also a lazy typist, so define some memorable abbreviations.

"Searches case insensitively if the pattern is all lowercase. Search case sensitively otherwise."

⟨*ripgrep fish abbreviations*⟩≡
```
rg = "rg -S";
```

⟨*ripgrep fish abbreviations*⟩+≡
```
rga = "rg -hidden -iglob !.git";
rgf = "rg -F";
```

"Never print the file path with the matched lines", and "[s]uppress line numbers."

⟨*ripgrep fish abbreviations*⟩+≡
```
rgin = "rg -IN";
```

```
λ rg -IN name: helmfile.d/*.yaml | sort -u
- name: cert-manager
- name: cluster-autoscaler
- name: elastalert
- name: elasticsearch-client
- name: elasticsearch-curator
- name: elasticsearch-exporter
- name: elasticsearch-master
- name: external-dns
- name: fluentd-elasticsearch
- name: hubble
- name: kibana
- name: kube-resource-report
- name: metrics-server
- name: nginx-ingress-core
- name: prometheus-operator
```

⟨*ripgrep fish abbreviations*⟩+≡
```
rgn = "rg -no-heading";
```

Define some shell aliases.

⟨*config/fish/aliases.nix*⟩≡
```
{ ... }:

{

  programs.fish.shellAliases = {
    ⟨Define some shell aliases⟩
  };

}
```

Help me replace autojump with zoxide.

⟨*Define some shell aliases*⟩≡
```
j = "z";
```

I've aliased `k` to `clear` for years and am too stubborn to change, even though all the cool kids alias `k` to `kubectl` these days.

⟨*Define some shell aliases*⟩+≡
```
k = "clear";
```

Define some short exa aliases. add link

⟨*Define some shell aliases*⟩+≡
```
l = "exa -color=auto -G";
ll = "exa -color=auto -Gla";
```

I liked `pbcopy` and `pbpaste` on Darwin, so I "ported" them to NixOS.

⟨*Define some shell aliases*⟩+≡
```
pbcopy = "xclip -sel clipboard";
pbpaste = "xclip -sel clipboard -o";
```

⟨*config/fish/shellInit.fish*⟩≡ % ⟹ this file was generated automatically by noweave — better not edit it

```
for p in /run/current-system/sw/bin ~/bin
    if not contains $p $fish_user_paths
        set -U fish_user_paths $p $fish_user_paths
    end
end

set -U fish_user_paths /run/wrappers/bin $fish_user_paths


function fish_title
  echo "$PWD | $_" | sed "s|$HOME|~|g"
end
```

⟨*config/fish/interactiveShellInit.fish*⟩≡ % ⟹ this file was generated automatically by noweave — better not edit it

```
function clone
  function __update
    test -d $argv[1]; and cd $argv[1]; and git fetch -all; and git pull
  end

  function __usage
    echo "Usage: clone [username] [repository] [[destination]]"
  end

  set -local num_args (count $argv)

  if test $num_args -ge 2
    set -local user $argv[1]
    set -local repo $argv[2]

    if test $num_args -eq 2
      set dest ~/src/$user/$repo
    else if test $num_args -eq 3
      set dest $argv[3]/$user/$repo
    else
      __usage
    end

    echo $dest

    git clone git@github.com:$user/$repo.git $dest; or __update $dest; or __usage
  else
    __usage
  end
end
```

⟨*config/fish/interactiveShellInit.fish*⟩+≡

```
function latest -d 'Print the latest release (on GitHub) for a given user and repo.'
  # TODO: __usage

  set -local num_args (count $argv)

  if test $num_args -eq 2
    set -local user $argv[1]
    set -local repo $argv[2]
    http https://api.github.com/repos/$user/$repo/releases/latest | jq -r '.tag_name'
  end
end
```

⟨*config/fish/interactiveShellInit.fish*⟩+≡
```
command -sq aws; and command -sq jq; and \
function describe-cert -d 'List the domains for a given ACM certificate'
    test (count $argv) -ne 1; and return
    aws acm describe-certificate --certificate-arn $argv[1] |
    jq -r '.Certificate | .SubjectAlternativeNames[]'
end
```

⟨*config/fish/interactiveShellInit.fish*⟩+≡
```
command -sq fluidsynth; and function playmidi
    fluidsynth -i ~/lib/arachno-soundfont/Arachno\ SoundFont\ -\ Version\ 1.0.sf2 $argv
end
```

⟨*config/fish/interactiveShellInit.fish*⟩+≡
```
command -sq kitty; and function icat
    kitty +kitten icat $argv
end
```

⟨*config/fish/interactiveShellInit.fish*⟩+≡
```
command -sq kubectl; and begin
    # TODO: Add option to print server versions too.
    function k8s::versions
        printf "kubectl %s\n" (command kubectl version --client --short)
        printf "helm %s\n" (command helm version --client --short)
        command helmfile --version
        printf "kops %s\n" (command kops version)
    end

    function kcterm -d 'Terminate a Kubernetes node'
        test (count $argv) -ne 1; and return
        kubectl get node -o jsonpath='{.spec.providerID}' $argv[1] |
        cut -d'/' -f5 |
        xargs aws ec2 terminate-instances --instance-ids
    end
end
```

⟨*config/fish/interactiveShellInit.fish*⟩+≡
```
# FIXME: functions rvm >/dev/null 2>&1; and rvm default
```

⟨*config/fish/interactiveShellInit.fish*⟩+≡
```
set fish_greeting
```

⟨*config/fish/interactiveShellInit.fish*⟩+≡
```
command -sq task; and command -sq jq; and function tj \
  -d 'Open the Jira ticket associated with a Taskwarrior task'
  test (count $argv) -ne 1; and return
  open (task $argv[1] export | jq -r '.[0].jiraurl')
end
```

fzf

⟨*config/fzf.nix*⟩≡

```
{ ... }:

{

  programs.fzf = {
    enable = true;
    enableZshIntegration = false;
  };

}
```

Git

⟨*config/git/default.nix*⟩≡ `% ⟹ this file was generated automatically by noweave — better not edit it`

```
{ config, ... }:

{
  imports = [
    ./aliases.nix
    ./config.nix
    ./lab.nix
    ./packages.nix
  ];

  programs.git = {
    enable = true;
    ignores = [
      "*~"
      ".DS_Store"
    ];
  };

}
```

⟨*config/git/packages.nix*⟩≡ `% ⟹ this file was generated automatically by noweave — better not edit it`

```
{ pkgs, ... }:

{
  home.packages = with pkgs; (
    [
      diff-pdf
      git
      github-cli
      kdiff3
      nix-prefetch-git
      nix-prefetch-github
      sops
    ] ++ (
      with gitAndTools; [
        git-extras
        gita
        hub
        lab
      ]
    )
  );

}
```

⟨*config/git/config.nix*⟩≡ `% ⟹ this file was generated automatically by noweave — better not edit it`

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

{

  programs.git = {
    delta = {
      enable = true;
      options = {
        plus-style = "syntax #012800";
        minus-style = "syntax #340001";
        syntax-theme = "Monokai Extended";
        navigate = true;
      };
    };

    extraConfig = {
      color = {
        diff-highlight = {
          oldNormal = "red bold";
          oldHighlight = "red bold 52";
          newNormal = "green bold";
          newHighlight = "green bold 22";
        };

        diff = {
          meta = 227;
          frag = "magenta bold";
          commit = "227 bold";
          old = "red bold";
          new = "green bold";
          whitespace = "red reverse";
        };

        status = {
          added = "green";
          changed = "yellow";
          untracked = "cyan";
        };
        ui = true;
      };

      commit.template = "${config.xdg.dataHome}/git/commit.template";

      credential = {
        helper = "${pkgs.gitAndTools.pass-git-helper}/bin/pass-git-helper";
        useHttpPath = true;
      };

      diff = {
        sopsdiffer = {
          textconv = "sops -d";
        };
      };

      difftool = {
        pdfdiffer.cmd = "diff-pdf -view \"$LOCAL\" \"$REMOTE\"";
```

```
      prompt = false;
      trustExitCode = true;
    };

    fetch.prune = true;

    init.defaultBranch = "main";

    pull.ff = "only";

    rerere.enabled = true;

    url."git@gitlab.sportradar.ag:" = {
      insteadOf = "https://gitlab.sportradar.ag/";
    };
  };

  includes =
    let
      inherit (config.accounts.email) accounts;
      personal = {
        signing.key =
          lib.optional
            (lib.hasAttrByPath [ "gpg" "key" ] accounts.personal)
            accounts.personal.gpg.key;
        user.email = accounts.personal.address;
      };
      work = {
        signing.key =
          lib.optional
            (lib.hasAttrByPath [ "gpg" "key" ] accounts.work)
            accounts.work.gpg.key;
        user.email = accounts.work.address;
      };
    in
    [
      {
        condition = "gitdir:~/src/git.sr.ht/";
        contents = personal;
      }
      {
        condition = "gitdir:~/src/github.com/";
        contents = personal;
      }
      {
        condition = "gitdir:~/src/gitlab.com/";
        contents = personal;
      }
      {
        condition = "gitdir:~/src/gitlab.sportradar.ag/";
        contents = work;
      }
    ];

  lfs.enable = true;

  userName =
```

```
      (builtins.head
        (lib.filter (account: account.primary)
          (lib.attrValues config.accounts.email.accounts))).realName;
  };

  xdg.configFile."pass-git-helper/git-pass-mapping.ini" = {
    source = ./git-pass-mapping.ini;
  };

  xdg.dataFile."git/commit.template" = {
    source = ./commit.template;
  };

}
```

Aliases

⟨*config/git/aliases.nix*⟩≡

```
{ config, lib, ... }:

{

  programs.fish.shellAbbrs = lib.mkIf (config.programs.fish.enable) {
    g = "git";
    ga = "gita";
    gaf = "gita fetch";
    gall = "gita ll";
    gd = "git diff";
    gdc = "git diff --cached";
    gm = "git merge";
    gs = "git status --short --untracked-files=no";
    gt = "git tree";
  };

  programs.git.aliases = rec {
    ap = "add --patch";
    bm = "branch --merged";
    bnm = "branch --no-merged";
    ca = "commit --amend";
    cam = "${ca} --message";
    can = "${ca} --no-edit";
    cann = "${can} --no-verify";
    cans = "${can} --gpg-sign";
    cas = "${ca} --gpg-sign";
    casm = "${cas} --message";
    cm = "commit --message";
    cnm = "commit --no-verify --message";
    co = "checkout";
    cob = "${co} -b";
    # FIXME: conflicts with git-extras
    # cp = "cherry-pick";
    cpa = "cherry-pick --abort";
    cpc = "cherry-pick --continue";
    cpm = "cherry-pick -xm1";
    cpx = "cherry-pick -x";
    csm = "commit --gpg-sign --message";
    d = "diff";
    dad = "add";
    dc = "diff --cached";
    ds = "diff --stat";
    # ffco = "flow feature checkout";
    # ffr = "flow feature rebase";
    # ffs = "flow feature start";
    # frf = "flow release finish";
    # frfs = "flow release finish -s";
    # frs = "flow release start";
    r = "reset";
    rb = "rebase";
    rba = "rebase --abort";
    rbc = "rebase --continue";
    rbi = "rebase --interactive";
    rbs = "rebase --skip";
```

```
      rest = "reset";
      rh = "reset --hard";
      sa = "stash apply";
      sk = "stash --keep-index";
      sl = "stash list";
      sp = "stash pop";
      spa = "stash --patch";
      ss = "stash save";
      st = "status --short";
      stu = "status --short --untracked-files=no";
      stat = "status";
      tree = "log --all --graph --oneline";
    };

  }
```

GPG

⟨*config/gpg.nix*⟩≡

```
  { config, lib, ... }:

  {

    programs.gpg = {
      enable = true;
      settings = {
        default-key =
          (builtins.head
            (lib.filter (account: account.primary)
              (lib.attrValues config.accounts.email.accounts))).gpg.key;
        keyid-format = "long";
        no-emit-version = true;
      };
    };

    services.gpg-agent = {
      enable = true;
      defaultCacheTtl = 28800;
      enableSshSupport = true;
      maxCacheTtl = 28800;
    };

  }
```

htop

⟨*config/htop.nix*⟩≡
```
{ ... }:

{

  programs.htop = {
    enable = true;
    settings = {
      color_scheme = 6;
      cpu_count_from_zero = true;
      highlight_base_name = true;
      show_cpu_usage = true;
    };
  };

}
```

jq

⟨*config/jq.nix*⟩≡
```
{ ... }:

{

  programs.jq.enable = true;

}
```

Keyboard

⟨*config/keyboard.nix*⟩≡
```
{ ... }:

{

  home.keyboard.options = [
    "compose:ralt"
    "ctrl:nocaps"
  ];

}
```

kitty

⟨*config/kitty.nix*⟩≡
```
{ lib, pkgs, ... }:

{

  programs.kitty = {
    enable = true;
    extraConfig = ''
      include theme.conf
    '';
    font = {
      name = "Iosevka Term";
      package = (pkgs.nerdfonts.override { fonts = [ "Iosevka" ]; });
    };
    keybindings = {
      "kitty_mod+enter" = "new_window_with_cwd";
      "kitty_mod+k" = ''
        combine : clear_terminal scrollback active : send_text normal \x0c
      '';
    };
    settings = {
      editor = "emacsclient -nw -a \"\"";
      font_size = lib.mkDefault 24;
      kitty_mod = "ctrl+shift";
      scrollback_lines = -1;
      shell = ".";
      term = "xterm";
      window_border_width = 0;
    };
  };

  ⟨Configure the theme⟩

}
```

Theme

Download (and patch) the Wombat theme[52].

[52] https://github.com/dexpota/kitty-themes/#wombat

⟨*Configure the theme*⟩≡
```
xdg.configFile."kitty/theme.conf".source = pkgs.fetchurl {
  url = "https://raw.githubusercontent.com/dexpota/kitty-themes/c4bee86c/themes/Wombat.conf";
  hash = "sha256-macm9bb/9zWLeFANXqiYPc5IS40A7ZbhXr/DooJARsQ=";
  postFetch = ''
    ⟨Tweak some colors⟩
  '';
};
```

Tweak some of the colors, based on Emacs's wombat theme.

⟨*Tweak some colors*⟩≡
```
${pkgs.gawk}/bin/gawk -i inplace '
    /^background/           { sub($2, "#242424") }
    /^foreground/           { sub($2, "#f6f3e8") }
    /^cursor/               { sub($2, "#656565") }
    /^selection_background/ { sub($2, "#444444") }
    /^color0/               { sub($2, "#242424") }
    /^color8/               { sub($2, "#303030") }
    /^selection_foreground/ { sub($2, "#f6f3e8") }
    { print }
' $out
```

man

⟨*config/man.nix*⟩≡
```
{ ... }:

{

  programs.man.enable = true;

}
```

⟨*config/password-store.nix*⟩≡ `% ⟹ this file was generated automatically by noweave — better not edit it`
```
{ config, pkgs, ... }:

{
  home.packages = with pkgs; [
    gitAndTools.pass-git-helper
  ];

  programs.password-store = {
    enable = true;
    package = pkgs.pass.withExtensions (exts: with exts; [
      pass-genphrase
      pass-otp
      pass-tomb
      pass-update
    ]);
    settings = {
      PASSWORD_STORE_DIR = "${config.home.homeDirectory}/.password-store";
    };
  };

}
```

rebar3

⟨*config/rebar3.nix*⟩≡

```
{ ... }:

{

  xdg.configFile."rebar3/rebar.config".text = ''
    {plugins, [rebar3_hex]}.
  '';

}
```

Starship

Starship[53] is the minimal, blazing-fast, and infinitely customizable prompt for any shell!

[53] https://starship.rs

⟨*config/starship.nix*⟩≡

```
{ ... }:
{
  programs.starship = {
    enable = true;
    settings = {
      add_newline = false;
      character = {
        success_symbol = "[λ](bold green)";
        error_symbol = "[λ](bold red)";
      };
      format = builtins.concatStringsSep "" [
        "$username"
        "$hostname"
        "$shlvl"
        "$kubernetes"
        "$directory"
        # "$vcsh"
        "$git_branch"
        "$git_commit"
        "$git_state"
        "$git_status"
        # "$hg_branch"
        # "$docker_context"
        "$package"
        # "$cmake"
        # "$dart"
        # "$deno"
        # "$dotnet"
        "$elixir"
        # "$elm"
        "$erlang"
        "$golang"
        "$helm"
        "$java"
        # "$julia"
        # "$kotlin"
        # "$nim"
        "$nodejs"
        "$ocaml"
        "$perl"
        # "$php"
        "$purescript"
        "$python"
        # "$red"
        "$ruby"
        "$rust"
        # "$scala"
        # "$swift"
        "$terraform"
        # "$vagrant"
        # "$zig"
```

```
        "$nix_shell"
        # "$conda"
        "$memory_usage"
        "$aws"
        # "$gcloud"
        # "$openstack"
        "$env_var"
        # "$crystal"
        # "$custom"
        "$cmd_duration"
        # "$line_break"
        # "$lua"
        "$jobs"
        # "$battery"
        "$time"
        "$line_break" # added
        "$status"
        # "$shell"
        "$character"
      ];
      git_branch.symbol = " ";
      git_commit.tag_disabled = false;
      git_status = {
        ahead = "⇡"${count}";
        behind = "⇣"${count}";
        diverged = "⇕ ⇡"${ahead_count}⇣"${behind_count}";
        staged = "+$count";
      };
      kubernetes.disabled = false;
      nix_shell = {
        format = "via [$symbol$state]($style) ";
        impure_msg = "ι";
        pure_msg = "ρ";
        symbol = "";
      };
      time.disabled = false;
    };
  };
}
```

Taskwarrior

Config

⟨*config/taskwarrior/default.nix*⟩≡

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

{
```

Install Timewarrior[54].

[54] https://timewarrior.net

⟨*config/taskwarrior/default.nix*⟩+≡

```
  home.packages = with pkgs; [
    timewarrior
  ];
```

Since home-manager generates a read-only `.taskrc` file, and using contexts[55] requires Taskwarrior to be able to modify the it, use the following hacky workaround.

Set `$TASKRC` to `~/.taskrc-dirty`, which must contain at least `include ~/.config/task/taskrc`.

[55] https://taskwarrior.org/docs/context.html

⟨*config/taskwarrior/default.nix*⟩+≡
```
home.sessionVariables = {
  TASKRC = "~/.taskrc-dirty";
};
```

Define short Taskwarrior shell aliases.

⟨*config/taskwarrior/default.nix*⟩+≡
```
programs.fish.shellAliases = lib.mkIf (config.programs.fish.enable) rec {
  p = "task ls limit:page";
  po = "timew summary :week";
  pp = tbd;
  t = "task limit:page";
  ta = "task add";
  tbd = "task burndown.daily";
  te = "env VISUAL=$EDITOR task edit";
  tl = "task list";
  tm = "task mod";
  tw = "timew";
};
```

Configure Taskwarrior, using home-manager[56].

[56] https://github.com/rycee/home-manager

⟨*config/taskwarrior/default.nix*⟩+≡
```
programs.taskwarrior = {
  enable = true;
  colorTheme = "solarized-dark-256";
  config = {
    context.other = "-work";
    context.work = "+work";
  };
};
```

Link the on-exit Git hook. [57]

[57] See next subssection.

⟨*config/taskwarrior/default.nix*⟩+≡
```
xdg.dataFile."task/hooks/on-exit-git.sh" = {
  executable = true;
  source = ./on-exit-git.sh;
};
```

⟨*config/taskwarrior/default.nix*⟩+≡

```
    xdg.dataFile."task/hooks/on-modify.timewarrior" = {
      executable = true;
      source = let inherit (pkgs.timewarrior) version; in
        with pkgs; stdenv.mkDerivation {
          pname = "taskwarrior-on-modify.timewarrior";
          inherit version;
          nativeBuildInputs = [ makeWrapper ];
          buildInputs = [ python3 ];
          src = fetchurl {
            url = "https://raw.githubusercontent.com/GothenburgBitFactory/timewarrior/v${version}/ext/on-modify.timew
            sha512 = "sha512-GsDqetyfQOUU+VTZbgdKH1X6n5tM7q3Q0B5X/zk+JHgzw6vVk48IxGvCaDnpIJXCASIgSGsKLxLvv7RPDdlPAw==
          };
          dontUnpack = true;
          installPhase = "
            install -m755 $src $out
            substituteInPlace $out -replace "/usr/bin/env " ${python3}/bin/
          ";
        };
    };

  }
```

on-exit Git hook

First, use a fairly portable Bash shebang, and be safe.[58]

[58] `-e`xit immediately upon failure, treat `-u`nset variables as an error, disable `-f`ile globbing, and fail if any part of a pipeline fails (`-o pipefail`).

⟨*config/taskwarrior/on-exit-git.sh*⟩≡

```
  #! /usr/bin/env bash

  set -eufo pipefail
```

Run in debug mode[59], if the environment variable `DEBUG` is nonempty.

[59] i.e. echo commands

⟨*config/taskwarrior/on-exit-git.sh*⟩+≡

```
  if [ -n "${DEBUG:-}" ]; then
      set -x
  fi
```

Parse the command line arguments.

Don't include the `api` version, `rc` file, or Taskwarrior `version`.

N.B. The positional arguments are formatted as `name:value`.

```
api="${1#api:}"
rc="${4#rc:}"
version="${6#version:}"
```

Parse and store the values of the arguments, `args`, `command`, and `data` directory.

⟨*config/taskwarrior/on-exit-git.sh*⟩+≡

```
  args="${2#args:}"
  command="${3#command:}"
  data="${5#data:}"
```

Throughout this script, run `git` as if it were started in the Taskwarrior `data` directory, i.e. `git -C "$data"`.

⟨*config/taskwarrior/on-exit-git.sh*⟩+≡
```
if git -C "$data" diff -quiet; then
    if [ -n "${DEBUG:-}" ]; then
        echo 'No changes to commit'
    fi
    exit 0
```

If there are no changes, `exit` successfully, printing an informative message if running in debug mode. `git diff` `exit`s with status code `0` if there are no differences between the working tree and the index.

If present, stage the changes or die trying.

⟨*config/taskwarrior/on-exit-git.sh*⟩+≡
```
elif ! git -C "$data" add -A; then
    echo 'Failed to add files to the index'
    exit 100
```

Try to commit the changes, or else.

⟨*config/taskwarrior/on-exit-git.sh*⟩+≡
```
elif ! git -C "$data" commit -qm "$command: ${args#task "$command"}"; then
    echo 'Failed to record changes to the repository'
    exit 101
```

Quietly store the current contents of the index in a new commit, along with a log message describing the changes.

If running in debug mode, print a brief summary of the commit.

⟨*config/taskwarrior/on-exit-git.sh*⟩+≡
```
elif [ -n "${DEBUG:-}" ]; then
    git -C "$data" log -oneline -1
fi
```

xmonad

⟨*config/xmonad/default.nix*⟩≡

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

{

  # FIXME
  # imports = [
  #   ../dunst.nix
  #   ../fonts.nix
  #   ../rofi.nix
  #   ../screen-locker.nix
  # ];

  home.packages = with pkgs; [
    flameshot
    # font-awesome_4
    haskellPackages.xmobar
    playerctl
    # FIXME: xorg.xbacklight
  ];

  home.pointerCursor = {
    package = pkgs.vanilla-dmz;
    name = "Vanilla-DMZ-AA";
    size = 36;
    x11.enable = true;
  };

  xdg.configFile."xmobar/xmobarrc" = {
    source = ../xmobar/xmobarrc;
    # NOTE: https://github.com/nix-community/home-manager/issues/1399
    onChange = "
      if [[ -v DISPLAY ]]; then
          echo "Restarting xmonad"
          $DRY_RUN_CMD ${config.xsession.windowManager.command} --restart
      fi
    ";
  };

  xsession = {
    enable = true;
    initExtra = lib.mkIf (config.targets.genericLinux.enable) "
      xrandr -s 1920x1200 --output eDP-1 --scale 2
      xmonad --restart
      systemctl --user restart picom.service random-background.service
    ";
    scriptPath = ".hm-xsession";
    windowManager.xmonad = {
      enable = true;
      extraPackages = hpkgs: [ hpkgs.xmonad-contrib ];
      config = ./xmonad.hs;
    };
  };

}
```

Glossary

module "Each NixOS module is a file that handles one logical aspect of the configuration, such as a specific kind of hardware, a service, or network settings."[60] 21

[60] https://nixos.org/nixos/manual/index.html#sec-writing-modules

Acronyms

LUKS Linux Unified Key Setup 21, 25

To-Do