

ERIC BAILEY

# DOTFILES



# Contents

## *Software configuration*      5

*Bash*      5

*bat*      5

*Browserpass*      7

*bugwarrior*      7

*direnv*      8

*Emacs*      8

*Init File*      11

*Firefox*      14

*fish*      15

*fzf*      25

*Git*      25

*GPG*      30

*htop*      30

*i3*      31

*i3status*      32

*jq*      33

*kitty*      33

*man*      34

*nixpkgs*      35

*rebar3*      35

*Taskwarrior*      35

*Config*      35

*on-exit Git hook*      38

## *Bibliography*      43



# Software configuration

## Bash

Enable the Bash module, as well as autojump.

`<config/bash.nix>≡`

```
{ ... }:  
  
{  
  
  programs.bash = {  
    enable = true;  
    enableAutojump = true;  
  };  
  
}
```

## bat

Enable the `bat`<sup>1</sup> module.

<sup>1</sup> <https://github.com/sharkdp/bat>

`<config/bat.nix>≡`

```
{ ... }:  
  
{
```

```
  programs.bat = {  
    enable = true;  
    config = {
```

For the pager, use `less`.

`<config/bat.nix>+≡`

```
    pager = "less -FR";
```

Set the style to `changes`.

`<config/bat.nix>+≡`

```
    style = "changes";
```

Use the Monokai Extended theme.

`<config/bat.nix>+≡`

```
    theme = "Monokai Extended";
```

$\langle config/bat.nix \rangle + \equiv$

```
    };  
  };  
}
```

*Browserpass**<config/browserpass.nix>*≡

```

{ pkgs, ... }:

{

  home.packages = with pkgs; [
    browserpass
  ];

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

}

```

*bugwarrior**<config/bugwarrior.nix>*≡

```

{ config, pkgs, ... }:

{

  home.packages = with pkgs; [
    bugwarrior
    pass
  ];

  imports = [
    ./taskwarrior
  ];

  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 = e.bailey and resolution = unresolved
    jira.version = 8
    jira.add_tags = work
    jira.description_template = {{jiraid}}: {{jirasummary}}

```

```
'';
}
```

### *direnv*

$\langle \text{config/direnv.nix} \rangle \equiv$

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

### *Emacs*

$\langle \text{config/emacs/default.nix} \rangle \equiv$

```
{ config, lib, pkgs, ... } :
{
  imports = [
    ./packages.nix
  ];

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

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

  programs.emacs.enable = true;

  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;
}
```



```

<config/emacs/packages.nix>≡
{ ... }:
{
  programs.emacs.extraPackages = epkgs: with {
    elpa = epkgs.elpaPackages;
    melpa = epkgs.melpaPackages;
    melpaStable = epkgs.melpaStablePackages;
    # org = epkgs.orgPackages;
  }; (
<config/emacs/packages.nix>+≡
    with elpa; [
      mmm-mode
    ]

```

`<config/emacs/packages.nix>+≡`

```
) ++ (
  with melpa; [
    avy
    better-defaults
    # clj-refactor
    # clojure-mode
    # TODO: company-lsp
    # TODO: cquery
    crux
    dhall-mode
    direnv
    dockerfile-mode
    editorconfig
    elixir-mode
    enh-ruby-mode
    # FIXME: erlang
    fill-column-indicator
    fish-mode
    gap-mode
    go-mode
    graphviz-dot-mode
    haskell-mode
    helm-ag
    # TODO: hindent
    hl-todo
    htmlize
    idris-mode
    j-mode
    kubernetes-tramp
    # TODO: lsp-haskell
    # TODO: lsp-mode
    # TODO: lsp-ui
    magit
    markdown-mode
    multiple-cursors
    nix-mode
    paredit
    rainbow-delimiters
    robe
    rust-mode
    rvm
    smex
    tuareg
    use-package
    whitespace-cleanup-mode
    yaml-mode
  ]
) ++ (
```

*<config/emacs/packages.nix>+≡*

```

    with melpaStable; [
      ess
    ]
  );
}

```

### *Init File*

*<config/emacs/init.el>≡*

```

(server-start)

<config/emacs/init.el>+≡
(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)

(tool-bar-mode 0)

<config/emacs/init.el>+≡
(let ((font (if (eq system-type 'darwin)
                "Iosevka-16"
                "Iosevka-10"))))
  (set-face-attribute 'default t :font font)
  (set-frame-font font nil t))

<config/emacs/init.el>+≡
(require 'package)

<config/emacs/init.el>+≡
(setq-default 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)

```

*<config/emacs/init.el>+≡*

*;; <https://stackoverflow.com/a/18330742>*

*<config/emacs/init.el>+≡*

```
(defvar --backup-directory
  (concat user-emacs-directory "backups"))
```

```
(unless (file-exists-p --backup-directory)
  (make-directory --backup-directory t))
```

```
(add-to-list 'backup-directory-alist `("." . --backup-directory))
```

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

*<config/emacs/init.el>+≡*

*<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>+≡*

```
(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>+≡*

```
(setq-default fill-column 80)
```

*<config/emacs/init.el>+≡*

```
(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 editorconfig
  :ensure t
  :config
  (editorconfig-mode 1))
```

*<config/emacs/init.el>+≡*

```
(use-package helm
  :demand
  :config
  (global-set-key (kbd "M-s-f") 'helm-do-grep-ag)
  (global-set-key (kbd "M-s-f") 'helm-do-grep-ag))
```

*<config/emacs/init.el>+≡*

```
(use-package hl-todo
  :demand
  :config (global-hl-todo-mode t))
```

*<config/emacs/init.el>+≡*

```
(use-package magit
  :demand
  :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 nix-mode
  :mode ("\\.nix\\'"))
```

*<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 robe
  :demand
  :mode ("\\.rb\\\""))
```

*<config/emacs/init.el>+≡*

```
(use-package rvm
  :ensure t)
```

*<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))
```

## Firefox

*<config/firefox.nix>≡*

```
{ pkgs, ... } :

{

  programs.firefox = {
    enable = true;
    extensions = with (import <nur> { inherit pkgs; }).repos.rycee.firefox-addons; [
      https-everywhere
      privacy-badger
    ];
  };
}
```

*fish*

$\langle \text{config/fish/default.nix} \rangle \equiv$

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

{

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

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

  programs.fish = let inherit (lib.strings) fileContents; in {
    enable = true;
    interactiveShellInit = fileContents ./interactiveShellInit.fish;
    promptInit = ''
      ${fileContents ./sushi/fish_prompt.fish}
      ${fileContents ./sushi/fish_right_prompt.fish}
    '';
    shellInit = fileContents ./shellInit.fish;
  };
}
```

```

<config/fish/abbrs.nix>≡
{ ... }:

{

  programs.fish.shellAbbrs = rec {
    # Kubernetes
    kc = "kubectl";
    kms = "kubens";
    kt = "kubetail";

    # Nix
    nb = "nix build";
    nbn = "${nb} --no-link";

    # ripgrep
    rga = "rg --hidden --iglob !.git";
    rgf = "rg -F";
    rgi = "rg -i";
    rgn = "rg --no-heading";
    rgs = "rg -S";

    # tree
    trea = "tree -a";
  };
}

<config/fish/aliases.nix>≡
{ ... }:

{

  programs.fish.shellAliases = rec {
    gpg = "gpg2";

    k = "clear";

    l = "ls -Glah";
    ll = "ls -Glh";
    ls = "ls -G";

    # Old Darwin habits
    pbcopy = "xclip -sel clipboard";
    pbpaste = "${pbcopy} -o";
  };
}

```



*<config/fish/shellInit.fish>≡*

```

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>≡*

```

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 tag (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/tags | jq '.[0].name'
```

```
  end
```

```
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
```

```
end
```

*<config/fish/interactiveShellInit.fish>+≡*

**command** -sq kubectl; **and begin**

**function** kcexec

argparse -N 2 --name=kcexec 'r/replica=' -- \$argv

**or return**

**if not set** -q \_flag-replica

**set** -l \_flag-replica 0

**end**

kubectl exec -it (kubectl get pod -o jsonpath="{.items[\$\_flag-replica].metadata.name}" -l app.kubernetes.io/n

**end**

**function** kcnodepods -d 'List all pods on a given node'

argparse --name=kcnodepods \

-N 1 -X 1 \

'n/namespace=?' \

'w/watch' \

-- \$argv

**or return**

**set** -l kubectl\_flags

**if set** -q \_flag-namespace

**set** kubectl\_flags \$kubectl\_flags --namespace=\$\_flag-namespace

**else**

**set** kubectl\_flags \$kubectl\_flags --all-namespaces

**end**

**if set** -q \_flag-watch

**set** kubectl\_flags \$kubectl\_flags --watch

**end**

**set** kubectl\_flags \$kubectl\_flags --field-selector=spec.nodeName=\$argv[1]

eval kubectl get pods \$kubectl\_flags

**end**

**function** cpo -d 'Get the name of the Cilium pod running on a given node'

**command** kubectl get pods \

--field-selector spec.nodeName=\$argv[1] \

--namespace kube-system \

--output jsonpath='{.items[0].metadata.name}' \

--selector k8s-app=cilium

**end**

**function** cpods -d 'Get the name of every Cilium pod'

**command** kubectl get pods \

--namespace kube-system \

--output jsonpath='{range .items[\*]}{.metadata.name}{ "\n"}' \

--selector k8s-app=cilium

**end**

*# 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
end

<config/fish/interactiveShellInit.fish>+≡

# FIXME: functions rvm >/dev/null 2>&1; and rvm default

<config/fish/interactiveShellInit.fish>+≡

if string match -r '.*k8s-\d+$' "$buildInputs"
    set fish_greeting (k8senv)
else
    set fish_greeting
end

<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

```

```
<config/fish/sushi/fish_prompt.fish>≡
```

```
function fish_prompt
  set -l code $status

  if set -q IN_NIX_SHELL
    set symbol "\n"
  else
    set symbol "\n"
  end

  if git::is_repo
    set -l branch (git::branch_name ^/dev/null)
    set -l ref (git show-ref --head --abbrev | awk '{print substr($0,0,7)}' | sed -n 1p)

    if git::is_stashed
      echo -n -s (white)"^"(off)
    end

    echo -n -s (red)" "(off)

    if git::is_dirty
      printf (white)"*" (off)
    end

    if command git symbolic-ref HEAD > /dev/null ^/dev/null
      if git::is_staged
        printf (cyan)"$branch" (off)
      else
        printf (yellow)"$branch" (off)
      end
    else
      printf (dim)"$ref" (off)
    end

    for remote in (git remote)
      set -l behind_count (echo (command git rev-list $branch..$remote/$branch ^/dev/null | wc -l | tr -d " "))
      set -l ahead_count (echo (command git rev-list $remote/$branch..$branch ^/dev/null | wc -l | tr -d " "))

      if test $ahead_count -ne 0; or test $behind_count -ne 0; and test (git remote | wc -l) -gt 1
        echo -n -s " "(orange)$remote(off)
      end

      if test $ahead_count -ne 0
        echo -n -s (white)" +"$ahead_count(off)
      end

      if test $behind_count -ne 0
        echo -n -s (white)" -$behind_count(off)
      end
    end

    echo -n -s (red)" "(off)
  end

  if test "$code" = 0
    echo -n -s (red)"$symbol" (off)
  else

```

```

        echo -n -s (dim)"$symbol"(off)
    end
end

<config/fish/sushi/fish_right_prompt.fish>≡

# ----- [ Colors ]

function orange
    set_color -o ee5819
end

function yellow
    set_color -o b58900
end

function red
    set_color -o d30102
end

function cyan
    set_color -o 2aa198
end

function white
    set_color -o fdf6e3
end

function dim
    set_color -o 4f4f4f
end

function off
    set_color -o normal
end

```

*<config/fish/sushi/fish\_right\_prompt.fish>+≡*

# ----- [ Git ]

```
function git::is_repo
    test -d .git; or command git rev-parse --git-dir >/dev/null ^/dev/null
end

function git::branch_name
    git::is_repo; and begin
        command git symbolic-ref --short HEAD ^/dev/null;
        or command git show-ref --head -s --abbrev | head -n1 ^/dev/null
    end
end

function git::is_dirty
    git::is_repo; and not command git diff --no-ext-diff --quiet --exit-code
end

function git::is_staged
    git::is_repo; and begin
        not command git diff --cached --no-ext-diff --quiet --exit-code
    end
end

function git::is_stashed
    git::is_repo; and begin
        command git rev-parse --verify --quiet refs/stash >/dev/null
    end
end
```

*<config/fish/sushi/fish\_right\_prompt.fish>+≡*

# ----- [ AWS ]

```
function aws::current_profile
    if set -q AWS_PROFILE
        printf "$AWS_PROFILE"
    else if set -q AWS_DEFAULT_PROFILE
        printf "$AWS_DEFAULT_PROFILE"
    end
end
```

```

<config/fish/sushi/fish_right_prompt.fish>+≡

# ----- [ Kubernetes ]

function k8s::current_context
    command kubectl config current-context
end

function k8s::current_namespace
    command kubectl config view --minify -o jsonpath='{.contexts[0].context.namespace}'
end

function k8s::current_user
    command kubectl config view --minify -o jsonpath='{.users[0].name}' \
        | string replace "@sportradar.com" ""
end

<config/fish/sushi/fish_right_prompt.fish>+≡

# ----- [ Right Prompt ]

function fish_right_prompt
    set -l cwd (basename (prompt_pwd))

    if git::is_repo
        set root_folder (command git rev-parse --show-toplevel ^/dev/null)
        set parent_root_folder (dirname $root_folder)
        set cwd (echo $PWD | sed -e "s|$parent_root_folder/||")
    end

    printf (yellow)"(" (dim)$cwd (yellow)" " (off)

    set -l aws_profile (aws::current_profile)
    test -n "$aws_profile";
    and printf (dim)"{" (yellow)(aws::current_profile)(dim)" " (off)

    command -sq kubectl; and begin
        if [ "e.bailey" = (k8s::current_user) ]
            printf (dim)"[" (off)
        else
            printf (red)"[" (off)
        end

        printf (yellow)(k8s::current_context)(off)

        set -l k8s_namespace (k8s::current_namespace)
        test -n "$k8s_namespace";
        and printf (dim)"/" (yellow)"$k8s_namespace" (off)

        if [ "e.bailey" = (k8s::current_user) ]
            printf (dim)"]" (off)
        else
            printf (red)"]" (off)
        end
    end

    printf (dim)(date +%H(yellow): (dim)%M(yellow): (dim)%S)(off)
end

```



*fzf*

$\langle \text{config/fzf.nix} \rangle \equiv$

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

*Git*

$\langle \text{config/git/default.nix} \rangle \equiv$

```
{ config, lib, pkgs, ... }:  
  
{  
  imports = [  
    ./aliases.nix  
    ./config.nix  
    ./packages.nix  
  ];  
  
  programs.git = {  
    enable = true;  
    ignores = [  
      "*~"  
      ".DS_Store"  
    ];  
    lfs.enable = true;  
  } // (  
    with config.accounts.email.accounts.primary; {  
      signing.key = gpg.key;  
      userEmail = address;  
      userName = realName;  
    }  
  );  
  
}
```

```

<config/git/packages.nix>≡
{ lib, pkgs, ... }:

{
  home.packages = with pkgs; (
    [
      diff-pdf
      git
      git-lfs
      github-cli
      kdiff3
      nix-prefetch-git
      nix-prefetch-github
      sops
    ] ++ (
      lib.optionals pkgs.stdenv.isLinux [
        # git-cola
      ]
    ) ++ (
      with gitAndTools; [
        git-crypt
        # gitflow
        hub
        lab
      ]
    ) ++ (
      with nodePackages; [
        codeowners
        diff-so-fancy
      ]
    )
  );
}

```

```

<config/git/config.nix>≡
{ config, ... }:

{

  programs.git.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";
      };
    };

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

    core.pager = "diff-so-fancy | less --tabs=4 -RFX";

    credential = {
      helper = "pass-git-helper";
      useHttpPath = true;
    };

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

    difftool = {
      kitty.cmd = "kitty +kitten diff $LOCAL $REMOTE";
      pdfdiffer.cmd = "diff-pdf --view \"$LOCAL\" \"$REMOTE\"";
      prompt = false;
      trustExitCode = true;
    };

    rerere.enabled = true;

    ui.color = true;
  };
}

```

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

```

<config/git/aliases.nix>≡
{ config, lib, ... }:

{

  programs.fish.shellAbbrs = lib.mkIf (config.programs.fish.enable) {
    g = "git";
    gd = "git d";
    gdc = "git dc";
    gs = "git st";
  };

  programs.git.aliases = rec {
    bm = "branch --merged";
    bnm = "branch --no-merged";
    ca = "commit --amend";
    cam = "${ca} -m";
    can = "${ca} --no-edit";
    cans = "${can} -S";
    cas = "${ca} -S";
    casm = "${cas} -m";
    cm = "commit -m";
    co = "checkout";
    cob = "${co} -b";
    cs = "crypt status";
    cse = "crypt status -e";
    csm = "commit -S -m";
    csu = "crypt status -u";
    d = "diff";
    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 -s";
    stat = "status";
    tree = "log --all --graph --oneline";
  };
}

```

*GPG* $\langle config/gpg.nix \rangle \equiv$ 

```

{ config, ... }:

{

  programs.gpg = {
    enable = true;
    settings = {
      default-key = config.accounts.email.accounts.primary.gpg.key;
      keyid-format = "long";
      no-emit-version = true;
    };
  };

  services.gpg-agent = {
    enable = true;
    defaultCacheTtl = 600;
    enableSshSupport = true;
    maxCacheTtl = 3600;
  };

}

```

*htop* $\langle config/htop.nix \rangle \equiv$ 

```

{ ... }:

{

  programs.htop.enable = true;

}

```

*i3*

`<config/i3/default.nix>≡`

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

{

  home.packages = with pkgs; [
    flameshot
    i3lock
    rofi
    rofi-pass
  ];

  services.compton.enable = true;

  xdg.configFile."i3status/config".source = ../i3status/config;

  xdg.dataFile."i3/matrix.png".source = ./matrix.png;

  xsession.enable = true;

  xsession.windowManager.i3 = {
    enable = true;
    config = let modifier = "Mod4"; in {
      fonts = [ "Iosevka 12" ];
      keybindings = lib.mkOptionDefault {
        "XF86MonBrightnessDown" = "exec xbacklight -dec 10";
        "XF86MonBrightnessUp" = "exec xbacklight -inc 10";
        "XF86AudioMute" = "exec --no-startup-id amixer set Master 1+ toggle";
        "XF86AudioLowerVolume" = "exec --no-startup-id amixer set Master 5%-";
        "XF86AudioRaiseVolume" = "exec --no-startup-id amixer set Master 5%+";
        "XF86AudioPrev" = "exec playerctl previous";
        # "XF86AudioPlayPause" = "exec playerctl play-pause";
        "XF86AudioPlay" = "exec playerctl play-pause";
        # "XF86AudioPause" = "exec playerctl play-pause";
        "XF86AudioNext" = "exec playerctl next";
        "${modifier}+Escape" = ''
          exec i3lock -i ${config.xdg.dataHome}/i3/matrix.png
        '';
        "${modifier}+Shift+minus" = "move scratchpad";
        "${modifier}+minus" = "scratchpad show";
        "${modifier}+Tab" = "exec --no-startup-id rofi -show window";
        "${modifier}+d" = ''
          exec --no-startup-id "rofi -combi-modi run,window,drun -show combi -modi combi"
        '';
        "${modifier}+space" = ''
          exec --no-startup-id "rofi -combi-modi run,window,drun -show combi -modi combi"
        '';
        "Print" = "exec flameshot gui";
      };
    };
    inherit modifier;
  };
};
```

*i3status*

```

<config/i3status/config>≡
general {
    colors = true
    interval = 5
    output_format = "i3bar"
}

order += "path_exists VPN"

path_exists VPN {
    format = "□"
    format_down = ""
    path = "/proc/sys/net/ipv4/conf/tun0"
}

order += "wireless _first_"

wireless _first_ {
    format_up = "%ssid"
}

order += "load"

load {
    format = "%1min %5min %15min"
    format_above_threshold = "□ %1min %5min %15min □"
    max_threshold = 9
}

order += "battery all"

battery all {
    format = "%status %percentage %remaining"
    low_threshold = 10
    status_bat = "□"
    status_chr = "%4"
    status_full = "□"
    status_unk = "□"
    threshold_type = "time"
}

order += "tztime trondheim"

tztime trondheim {
    format = "□□ %a %H:%M"
    hide_if_equals_localtime = true
    locale = "nb_NO.UTF-8"
    timezone = "Europe/Oslo"
}

order += "tztime local"

tztime local {
    format = "□ %a %d.%m %H:%M:%S"
}

```



*jq*

$\langle config/jq.nix \rangle \equiv$

```
{ ... }:  
  
{  
  
  programs.jq.enable = true;  
  
}
```

*kitty*

$\langle config/kitty/default.nix \rangle \equiv$

```
{ ... }:  
  
{  
  
  xdg.configFile."kitty/kitty.conf".source = ./kitty.conf;  
  
  xdg.configFile."kitty/diff.conf".text = ''  
    pygments_style monokai  
  '';  
  
}
```

$\langle config/kitty/kitty.conf \rangle \equiv$

```
font_family Iosevka  
bold_font Iosevka Medium  
italic_font Iosevka Italic  
bold_italic_font Iosevka Medium Italic
```

$\langle config/kitty/kitty.conf \rangle + \equiv$

```
font_size 12.0  
scrollback_lines -1
```

*<config/kitty/kitty.conf>+≡*

```
foreground #839496
background #002b36
```

```
color0 #073642
color1 #dc322f
color2 #859900
color3 #b58900
color4 #268bd2
color5 #d33682
color6 #2aa198
color7 #eee8d5
color8 #002b36
color9 #cb4b16
color10 #586e75
color11 #657b83
color12 #839496
color13 #6c71c4
color14 #93a1a1
color15 #fdf6e3
```

*<config/kitty/kitty.conf>+≡*

```
editor emacsclient -nw -a ""
```

*<config/kitty/kitty.conf>+≡*

```
term xterm
```

*<config/kitty/kitty.conf>+≡*

```
macos_option_as_alt yes
```

```
kitty_mod ctrl+shift
```

```
map kitty_mod+enter new_window_with_cwd
```

```
map kitty_mod+k combine : clear_terminal scrollback active : send_text normal \x0c
```

*man*

*<config/man.nix>≡*

```
{ ... }:
```

```
{
```

```
  programs.man.enable = true;
```

```
}
```

*nixpkgs*

$\langle \text{config/nixpkgs/default.nix} \rangle \equiv$

```
{ ... }:  
  
{  
  
  nixpkgs.config = import ./nixpkgs-config.nix;  
  
  xdg.configFile."nixpkgs/config.nix".source = ./nixpkgs-config.nix;  
  
}
```

$\langle \text{config/nixpkgs/nixpkgs-config.nix} \rangle \equiv$

```
{  
  allowBroken = false;  
  allowUnfree = true;  
  allowUnsupportedSystem = false;  
}
```

*rebar3*

$\langle \text{config/rebar3.nix} \rangle \equiv$

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

*Taskwarrior**Config*

$\langle \text{config/taskwarrior/default.nix} \rangle \equiv$

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

SINCE HOME-MANAGER GENERATES A READ-ONLY `.taskrc` FILE, and using `contexts`<sup>2</sup> requires Taskwarrior to be able to modify the it, use the following hacky workaround.

<sup>2</sup> <https://taskwarrior.org/docs/context.html>

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

`<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";
  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";
};
```

CONFIGURE TASKWARRIOR, using `home-manager`<sup>3</sup>.  
`<config/taskwarrior/default.nix>+≡`

<sup>3</sup> <https://github.com/rycee/home-manager>

```
programs.taskwarrior = {
  enable = true;
  colorTheme = "solarized-dark-256";
  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";
      };
      jiraaid = {
        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";
      };
    };
  };
};
```

LINK the on-exit Git hook.

ref section

```
<config/taskwarrior/default.nix>+≡

xdg.dataFile."task/hooks/on-exit-git.sh" = {
  executable = true;
  source = ./on-exit-git.sh;
};

}
```

*on-exit Git hook*

FIRST, use a fairly portable Bash shebang, and be safe.<sup>4</sup>

```
<config/taskwarrior/on-exit-git.sh>+≡
```

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

```
set -eufo pipefail
```

Run in debug mode<sup>5</sup>, if the environment variable `DEBUG` is nonempty.

<sup>4</sup> `-e` exit immediately upon failure, treat `-U` unset variables as an error, disable `-f` file globbing, and fail if any part of a pipeline fails (`-o pipefail`).

<sup>5</sup> 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 `foo:bar`, where `foo` is the name of the argument, and `bar` is the 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` `exits` 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
```










## *Bibliography*

Rafael J. Wysocki. CPU Performance Scaling – The Linux Kernel documentation. URL <https://www.kernel.org/doc/html/v4.19/admin-guide/pm/cpufreq.html#cpu-performance-scaling-in-linux>.



*To-Do*

 <a href="#">ref section</a> . . . . .	38
---	----