My Literate Emacs Configuration

Louis Ledoux

December 29, 2025

Contents

1 Initialization

This document contains a literate Emacs configuration. The prose explains **what** and **why**, while the tangled Emacs Lisp retains concise operational comments.

1.1 Package bootstrap

This section must run first. It disables implicit startup initialization, configures package archives, and ensures that `use-package` is available before any other configuration.

1.1.1 Disable implicit startup initialization

```
(setq package-enable-at-startup nil)   ;; prevent implicit init from early-init
```

1.1.2 Package system and archives

```
(require 'package)

(setq package-archives
      '(("gnu"   . "https://elpa.gnu.org/packages/")
        ("melpa" . "https://melpa.org/packages/")))
```

1.1.3 Initialization and archive refresh

```
(unless package--initialized
  (package-initialize))

(unless package-archive-contents
  (package-refresh-contents))
```

1.1.4 use-package bootstrap

`use-package` is used throughout the configuration to keep package declarations declarative and composable.

```
(unless (package-installed-p 'use-package)
  (package-install 'use-package))

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

(setq use-package-always-ensure t)   ;; auto-install dependencies
```

2 Core Emacs Behavior

This section configures baseline Emacs behavior unrelated to specific packages.

2.1 Startup and UI

2.1.1 Initial buffer state

```
(setq inhibit-startup-message t)   ;; start with empty screen
```

2.1.2 Disable UI chrome

```
(scroll-bar-mode -1)        ;; disable visible scrollbar
(tool-bar-mode -1)          ;; disable toolbar
(tooltip-mode -1)           ;; disable tooltips
(menu-bar-mode -1)          ;; disable menu bar
(set-fringe-mode 10)        ;; breathing room
```

2.1.3 Visual feedback

```
(setq visible-bell t)       ;; use visible bell instead of sound
```

2.2 Navigation and layout

2.2.1 Line numbers

```
(global-display-line-numbers-mode 1)
(setq display-line-numbers-type 'relative)   ;; relative numbers
```

2.2.2 Window splitting policy

```
(setq split-height-threshold nil
      split-width-threshold 0)   ;; split vertically by default
```

2.2.3 Escape behavior

```
(global-set-key (kbd "<escape>") #'keyboard-escape-quit)   ;; ESC quits prompts
```

2.3 Typography

2.3.1 Default font (optional)

```
;; (set-face-attribute 'default nil :font "Iosevka" :height 280)
(set-face-attribute 'default nil
                    :font "PxPlus IBM VGA8"
                    :height 280)
```

3 Appearance

3.1 Modus themes

The Modus themes provide a carefully designed, accessible color palette with light and dark variants.

3.1.1 Load local theme repository

```
(add-to-list 'load-path "~/.emacs.d/modus-themes")
(require 'modus-themes)
```

3.1.2 Active theme

```
(load-theme 'modus-operandi-tinted :no-confirm)
```

3.1.3 Theme toggling

```
(setq modus-themes-to-toggle
      '(modus-operandi-tinted modus-vivendi-tinted))

(define-key global-map (kbd "<f5>") #'modus-themes-toggle)
```

4 Minibuffer and Completion

This stack follows the modern Vertico + Orderless + Consult architecture, keeping Emacs completion native and extensible.

4.1 Core minibuffer semantics

4.1.1 Recursive minibuffers

```
(setq enable-recursive-minibuffers t)   ;; allow nested minibuffers
```

4.1.2 Command filtering

```
(setq read-extended-command-predicate
      #'command-completion-default-include-p)
```

4.1.3 Multiple selection indicator

This adds a visible indicator when completing multiple values.

```
(defun crm-indicator (args)
  (cons (format "[CRM%s] %s"
                (replace-regexp-in-string
                 "\\`\\[.*?]\\*\\|\\[.*?]\\*\\'" ""
                 crm-separator)
                (car args))
        (cdr args)))

(advice-add #'completing-read-multiple
            :filter-args #'crm-indicator)
```

4.1.4 Minibuffer prompt protection

```
(setq minibuffer-prompt-properties
      '(read-only t cursor-intangible t face minibuffer-prompt))
```

```
(add-hook 'minibuffer-setup-hook #'cursor-intangible-mode)
```

4.2 Completion UI

4.2.1 Vertico

Vertico provides a vertical minibuffer layout with eager candidate display.

```
(use-package vertico
  :config
  (setq vertico-cycle t        ;; cycle at boundaries
        vertico-count 30       ;; number of visible candidates
        vertico-resize nil)    ;; stable minibuffer height
  (vertico-mode 1))
```

4.2.2 Marginalia

Marginalia adds contextual annotations to completion candidates.

```
(use-package marginalia
  :config
  (marginalia-mode 1))
```

4.2.3 Orderless

Orderless enables out-of-order pattern matching using space- separated components.

```
(use-package orderless
  :custom
  (completion-styles '(orderless basic))
  (completion-category-defaults nil)
  (completion-category-overrides
   '((file (styles partial-completion)))))
```

4.2.4 Consult

Consult provides enhanced variants of standard search commands.

```
(use-package consult
  :bind (("M-s M-g" . consult-grep)
         ("M-s M-f" . consult-find)
         ("M-s M-o" . consult-outline)
         ("M-s M-l" . consult-line)
         ("M-s M-b" . consult-buffer)))
```

4.3 Actions and export

4.3.1 Embark

Embark exposes contextual actions for minibuffer candidates and buffer objects.

```
(use-package embark
  :bind (("C-." . embark-act)
         :map minibuffer-local-map
         ("C-c C-c" . embark-collect)
         ("C-c C-e" . embark-export)))
```

4.3.2 Embark–Consult integration

```
(use-package embark-consult)
```

4.3.3 Writable grep buffers

```
(use-package wgrep
  :bind (:map grep-mode-map
              ("e" . wgrep-change-to-wgrep-mode)
              ("C-x C-q" . wgrep-change-to-wgrep-mode)
              ("C-c C-c" . wgrep-finish-edit)))
```

5 State Persistence

5.1 Minibuffer history

```
(use-package savehist
  :init
  (savehist-mode 1))
```

5.2 Recently visited files

```
(recentf-mode 1)
```

6 Language and Editing Support

6.1 LaTeX

AUCTeX is used to ensure all ‘.tex‘ files open in LaTeX mode.

```
(use-package tex
  :ensure auctex
  :mode ("\\.tex\\'" . LaTeX-mode)
  :init
  (setq TeX-default-mode 'LaTeX-mode
        TeX-force-default-mode t     ;; do not guess from content
        TeX-parse-self t))
```

6.2 Visual assistance

6.2.1 Color literals

```
(use-package colorful-mode
  :hook (prog-mode text-mode LaTeX-mode))
```

6.2.2 Delimiter depth

```
(use-package rainbow-delimiters
  :hook ((prog-mode . rainbow-delimiters-mode)
         (latex-mode . rainbow-delimiters-mode)))
```

7 Discoverability

7.1 Keybinding hints

```
(use-package which-key
  :init
  (which-key-mode 1)
  :config
  (setq which-key-idle-delay 0.5
        which-key-idle-secondary-delay 0.1))
```

8 Org Infrastructure

8.1 Babel languages

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

9 Modal Editing

9.1 Evil

Evil provides Vim-style modal editing with Emacs integration.

```
(use-package evil
  :init
  (setq evil-want-integration t
        evil-want-keybinding nil
        evil-want-C-u-scroll t
        evil-want-C-i-jump nil
        evil-symbol-word-search t)
  :config
  (evil-mode 1))
```

9.1.1 Insert-state ergonomics

```
(define-key evil-insert-state-map
            (kbd "C-g") #'evil-normal-state)

(define-key evil-insert-state-map
            (kbd "C-h") #'evil-delete-backward-char-and-join)
```

9.1.2 Motion semantics

```
(evil-global-set-key 'motion "j" #'evil-next-visual-line)
(evil-global-set-key 'motion "k" #'evil-previous-visual-line)
```

9.1.3 Initial states

```
(evil-set-initial-state 'messages-buffer-mode 'normal)
(evil-set-initial-state 'dashboard-mode 'normal)
```

9.2 Evil collection

```
(use-package evil-collection
  :after evil
  :config
  (evil-collection-init))
```

10 Markup Formats

10.1 Markdown

```
(use-package markdown-mode
  :mode (("\\.md\\'" . markdown-mode)
         ("\\.markdown\\'" . markdown-mode))
  :init
  (setq markdown-command "markdown"))
```

11 Customization Backend

11.1 Custom variables

```
(custom-set-variables
 '(package-selected-packages nil))
```

11.2 Custom faces

```
(custom-set-faces)
```