Configuring Github Copilot in GNU Emacs

October 24, 2024

Ben Kirk

CISL / CSG

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Audience

Experienced Emacs users with at least some familiarity customizing your Emacs configuration.

Disclaimer

I'm not an Emacs developer. It's been my primary editor for 25+ years but I'm still reluctant (scared??) to change without a compelling reason. There's usually at least 5 ways to do anything in Emacs, I have no intention of learning or supporting them all.

What follows is a minimal configuration necessary to get Emacs working with a registered Github Copilot account. Especially if you have a complex customization already, your mileage may vary!

Roadmap

1. Configure the package installer `straight.el`,
2. Install `copilot-emacs` & `copilot-chat`,
3. Initial setup and login,
4. Simple usage.

Warning: Back up any pre-existing configuration first!!

```
# save your config files and directory:
$ cp ~/.emacs ~/.emacs.pre-copilot
$ cp -r ~/.emacs.d/ ~/.emacs.d.pre-copilot/
```

Roadmap

1. Configure the package installer `straight.el`,
2. Install `copilot-emacs` & `copilot-chat`,
3. Initial setup and login,
4. Simple usage.

Quickstart for the impatient:

```
# configuration for steps (1) & (2):
$ cat ~benkirk/dot.emacs-copilot >> ~/.emacs

# This will take a couple minutes to configure
# the new packages
$ emacs
  M-x copilot-install-server
  M-x copilot-login
```

`straight.el`: Installs Emacs packages from `git` repositories

- We will use **`straight.el`** as our base package manager for installing **`copilot-emacs`** & **`copilot-chat`**
  - Other installation methods for these tools are available, but this is the one I got working first.
- See the **`straight.el`** **getting started** page

- Dependencies:
  - **`git`**, Emacs ≥25.1

`~/.emacs` snippet:

```
;-----------------------------------------------
; 1: boostrap straight.el
; https://github.com/radian-software/straight.el#getting-started
(defvar bootstrap-version)
(let ((bootstrap-file
       (expand-file-name
        "straight/repos/straight.el/bootstrap.el"
        (or (bound-and-true-p straight-base-dir)
            user-emacs-directory)))
      (bootstrap-version 7))
  (unless (file-exists-p bootstrap-file)
    (with-current-buffer
        (url-retrieve-synchronously
         "https://raw.githubusercontent.com/radian-software/straight.el/develop/install.el"
         'silent 'inhibit-cookies)
      (goto-char (point-max))
      (eval-print-last-sexp)))
  (load bootstrap-file nil 'nomessage))
(setq package-enable-at-startup nil)
```

copilot-emacs & copilot-chat

- **copilot-emacs**: autocomplete on steroids.
- **copilot-chat**: Emacs / ChatGPT-style integration.
- Dependencies:
  - **straight.el**, Emacs ≥27, **node.js**, **npm**

~/.emacs snippet:

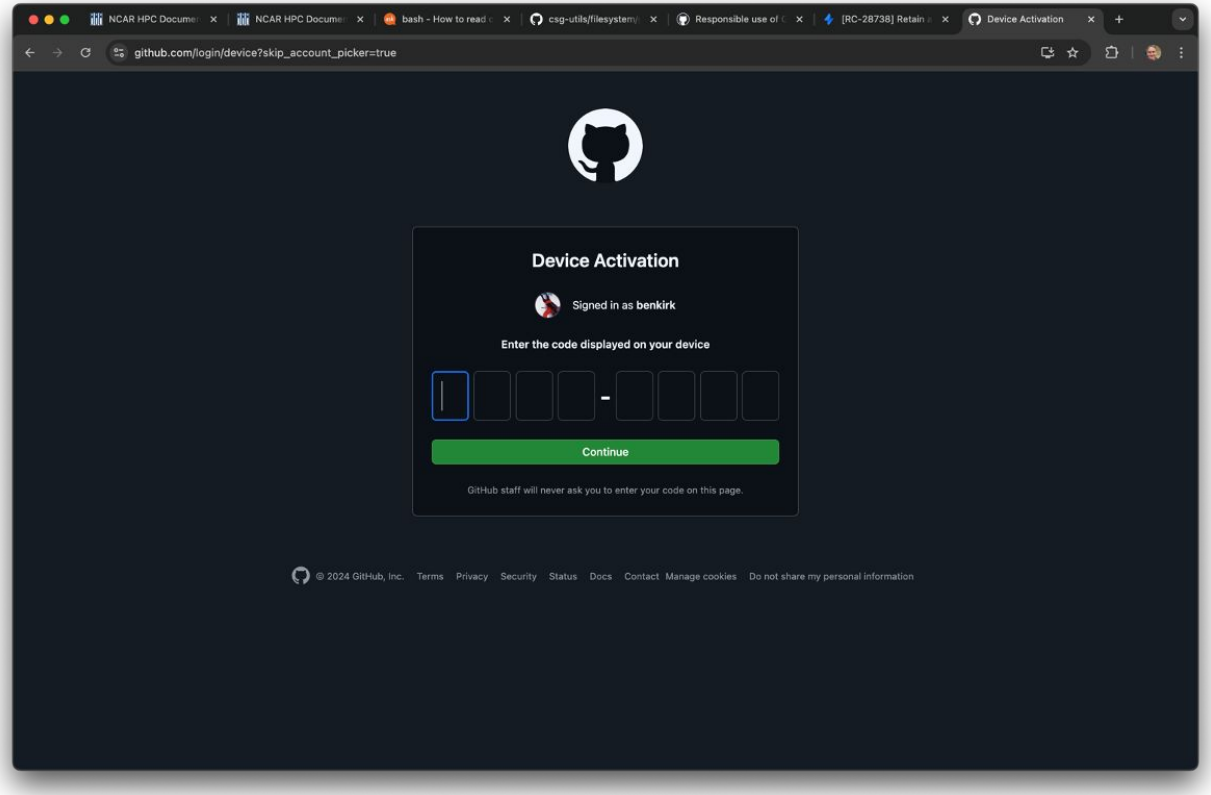
```
;----------------------------------------------
; 2: copilot-emacs
; https://github.com/copilot-emacs/copilot.el#installation
(use-package copilot
  :straight (:host github :repo "copilot-emacs/copilot.el" :files ("*.el"))
  :ensure t)
(require 'copilot)


;----------------------------------------------
; 3: copilot-chat
; https://github.com/chep/copilot-chat.el
(use-package copilot-chat
  :straight (:host github :repo "chep/copilot-chat.el" :files ("*.el"))
  :after (request))
```

Final Setup

1. Launch emacs,
   - After importing the previous config, this will take 1-2 minutes for first-time setup tasks

2. (once) Run
   - **`M-x copilot-install-server`**
   - **`M-x copilot-login`**
   - Navigate to `https://github.com/login/device` in a browser and enter the access code shown.
3. **`M-x copilot-mode`** to enable in a buffer



Demo

Common Functions

`copilot-mode`	Enables copilot-emacs auto completion for the current buffer.

`copilot-next-completion`
`copilot-accept-completion`
`[...]`
	Cycle through autocomplete options, etc…
	
	see https://robert.kra.hn/posts/2023-02-22-copilot-emacs-setup for customizing key bindings

`copilot-chat-explain`
`copilot-chat-fix`
`copilot-chat-review`
`copilot-chat-doc`
`copilot-chat-optimize`
`copilot-chat-test`
	For the selected code, ask Copilot to explain / fix / review etc…

`copilot-chat-display`
`copilot-chat-prompt`
`copilot-chat-prompt-send`
	Access a “prompt buffer” directly and send its contents as a query to Copilot

My General Impressions

Ben's overly-opinionated impressions:

- I personally don't much care for the autocomplete. I have used the built-in completion for years and find that sufficient - and works great offline too.
- I find the **`copilot-chat-{review,doc,explain,test}`** functions useful, so long as you have a healthy skepticism for the output.
- I do like **`copilot-chat-display`** simply because it allows me to stay in the editor vs. bouncing around to a web browser (and associated distractions).

Startup can be slower, so it's good to know about **`emacs -Q`** for rapid editing.

This same setup works on my Mac, which is typically where I run my editor for involved sessions.

- I find this gives me a better GUI experience.
- I use **TRAMP mode** for editing files remotely through a **`ssh`** connection.
- Let me know if this sounds interesting to you, but that's another talk.

Additional Resources

Required ~/.emacs configuration

```
;----------------------------------------------
; 1: boostrap straight.el
; https://github.com/radian-software/straight.el#getting-started
(defvar bootstrap-version)
(let ((bootstrap-file
       (expand-file-name
        "straight/repos/straight.el/bootstrap.el"
        (or (bound-and-true-p straight-base-dir)
            user-emacs-directory)))
      (bootstrap-version 7))
  (unless (file-exists-p bootstrap-file)
    (with-current-buffer
        (url-retrieve-synchronously
         "https://raw.githubusercontent.com/radian-software/straight.el/develop/install.el"
         'silent 'inhibit-cookies)
      (goto-char (point-max))
      (eval-print-last-sexp)))
  (load bootstrap-file nil 'nomessage))
(setq package-enable-at-startup nil)

;----------------------------------------------
; 2: copilot-emacs
; https://github.com/copilot-emacs/copilot.el#installation
(use-package copilot
  :straight (:host github :repo "copilot-emacs/copilot.el" :files ("*.el"))
  :ensure t)
(require 'copilot)

;----------------------------------------------
; 3: copilot-chat
; https://github.com/chep/copilot-chat.el
(use-package copilot-chat
  :straight (:host github :repo "chep/copilot-chat.el" :files ("*.el"))
  :after (request))
```

Copilot-aware Functions and Custom Key Bindings

```
*Completions*
 :straight (:host github :repo "copilot-emacs/copilot.el" :files ("*.el"))
 :ensure t)
(require 'copilot)
U:---  mac.emacs     74%   L98   Git:master  (Lisp +1) [71%]
 :ensure t)
(require 'copilot)

; https://robert.kra.hn/posts/2023-02-22-copilot-emacs-setup/
(defun rk/copilot-complete-or-accept ()
  "Command that either triggers a completion or accepts one if one
is available."
  (interactive)
  (if (copilot--overlay-visible)
      (progn
        (copilot-accept-completion)
        (open-line 1)
        (next-line))
    (copilot-complete)))

(define-key copilot-mode-map (kbd "M-C-<next>") #'copilot-next-completion)
(define-key copilot-mode-map (kbd "M-C-<prior>") #'copilot-previous-completion)
(define-key copilot-mode-map (kbd "M-C-<right>") #'copilot-accept-completion-by-word)
(define-key copilot-mode-map (kbd "M-C-<down>") #'copilot-accept-completion-by-line)
(define-key copilot-mode-map (kbd "M-<tab>") #'copilot-accept-completion)
(define-key global-map (kbd "M-C-<return>") #'rk/copilot-complete-or-accept)


;---------------------------------------------------
; https://github.com/chep/copilot-chat.el
(use-package copilot-chat
  :straight (:host github :repo "chep/copilot-chat.el" :files ("*.el"))
  :after (request))


;; Local Variables:
U:---  mac.emacs     76%   L101  Git:master  (Lisp +1) [71%]
In this buffer, type RET to select the completion near point.

41 possible completions:
copilot-accept-completion            copilot-accept-completion-by-line    copilot-accept-completion-by-paragraph
copilot-accept-completion-by-word    copilot-chat                         copilot-chat-add-current-buffer
copilot-chat-ask-and-insert          copilot-chat-custom-prompt-selection copilot-chat-del-current-buffer
copilot-chat-display                 copilot-chat-doc                     copilot-chat-explain
copilot-chat-fix                     copilot-chat-list                    copilot-chat-list-add-or-remove-buffer
copilot-chat-list-clear-buffers      copilot-chat-list-mode               copilot-chat-list-refresh
copilot-chat-mode                    copilot-chat-optimize                copilot-chat-prompt
copilot-chat-prompt-history-next     copilot-chat-prompt-history-previous copilot-chat-prompt-mode
copilot-chat-prompt-send             copilot-chat-prompt-split-and-list   copilot-chat-reset
copilot-chat-review                  copilot-chat-test                    copilot-clear-overlay
copilot-complete                     copilot-diagnose                     copilot-install-server
copilot-login                        copilot-logout                       copilot-mode
copilot-next-completion              copilot-panel-complete               copilot-previous-completion
copilot-reinstall-server             copilot-uninstall-server
U:%*-  *Completions*   Bot   L14    (Completion List +1) [71%]
M-x copilot-
```