Building Neovim Plugins: A Journey from Novice to Pro

Abhishek Keshri aka 2KAbhishek

GitHub (https://github.com/2kabhishek) | X (https://x.com/2kabhishek)

- Tech Lead at Incubyte
- From Rampurhat, West Bengal, India
- Love FOSS and the terminal

My (Neo)Vim Journey

- Vim (2018-2021) -> Neovim (2021-)

Plugins I've Built

- co-author.nvim: Add git co-authors
- nerdy.nvim: Nerd glyph finder
- tdo.nvim: Notes and todos
- termim.nvim: Improved terminal integration
- markit.nvim: Simpler marks
- octohub.nvim: Manage GitHub repos easily
- exercism.nvim: Exercism.io integration

Why Plugin Development?

- **Boost Productivity**: Automate tasks and streamline workflows.
- **Make Your Own Tools**: Create personalized commands and shortcuts.
- **Enhance Integration**: Connect seamlessly with other tools.
- **Empower the Community**: Share innovations and contribute to Neovim's growth.

Plugin Development: Pre-requisites

- Neovim!
- An idea
  - Anything that interrupts your flow state
  - Any useful functions part of your configs
- Functional Lua knowledge (`:h lua-guide`)
- Lazy.nvim
- A ready to use plugin template

> *Think about a plugin that you want to build*

Your First Plugin

Lets start with the basics

- Add commands to perform tasks
- Add keymaps to trigger commands quickly
- Add user configuration to customize behavior

Add Commands

```
vim.api.nvim_create_user_command('OctoRepos', function(opts)
    local user_arg, sort_arg, type_arg = '', '', ''

    for _, arg in ipairs(vim.split(opts.args, ' ')) do
        if arg:match('^sort:') then
            sort_arg = arg:sub(6)
        elseif arg:match('^type:') then
            type_arg = arg:sub(6)
        else
            user_arg = arg
        end
    end

    repos.show_repos(user_arg, sort_arg, type_arg)
end, { nargs = '*' })
```

> *From commands.lua in octohub.nvim*

Add Keymaps

```lua
if config.add_default_keybindings then
    local function add_keymap(keys, cmd, desc)
        vim.api.nvim_set_keymap('n', keys, cmd, { noremap = true, silent = true, desc = desc })
    end

    add_keymap('<leader>goo', ':OctoRepos<CR>', 'All Repos')
    add_keymap('<leader>gos', ':OctoRepos sort:stars<CR>', 'Top Starred Repos')
    add_keymap('<leader>goi', ':OctoRepos sort:issues<CR>', 'Repos With Issues')
    add_keymap('<leader>gou', ':OctoRepos sort:updated<CR>', 'Recently Updated Repos')
    add_keymap('<leader>gop', ':OctoRepos type:private<CR>', 'Private Repos')
    add_keymap('<leader>gof', ':OctoRepos type:fork sort:issues<CR>', 'Forked Repos')
end
```

From commands.lua in octohub.nvim

Add Configurations

```lua
---@class octohub.config
---@field sort_repos_by string : Sort repositories by various params
---@field repo_type string : Type of repositories to display
---@field repo_cache_timeout number : Time in seconds to cache repositories
---@field add_default_keybindings boolean : Whether to add default keybindings
local config = {
    sort_repos_by = '',
    repo_type = '',
    repo_cache_timeout = 3600 * 24 * 7,
    add_default_keybindings = true,
}

---@type octohub.config
M.config = config

---@param args octohub.config
M.setup = function(args)
    M.config = vim.tbl_deep_extend('force', M.config, args or {})
end
```

From config.lua in octohub.nvim

Overview of a Plugin's Structure

```
□ template.nvim
├── □ doc
│  └── □ template.txt    <- vimdoc, visible with :help
├── □ lua
│  ├── □ template
│  │  ├── □ commands.lua <- commands and keymaps
│  │  ├── □ config.lua   <- user configuration
│  │  └── □ module.lua   <- lua modules
│  └── □ template.lua    <- plugin entry point
├── □ tests
│  ├── □ init.lua        <- test setup
│  └── □ module_spec.lua <- module tests
├── □ .github
│  └── □ workflows
│      ├── □ ci.yml       <- lint and test
│      └── □ docs.yml     <- docs generation
├── □ Makefile           <- quick commands
├── □ README.md          <- plugin documentation
└── □ .stylua.toml       <- lua formatter config
```

> *From template.nvim (https://github.com/2kabhishek/template.nvim)*

Building Advanced Plugins

- Asynchronous APIs (for background tasks)
- Integration with external tools (Git, Docker, etc.)
- Building robust UIs within Neovim
- Using Treesitter and LSP for powerful editing capabilities
- Anything you can imagine!

Useful Plugins

- plenary.nvim
- utils.nvim
- nui.nvim
- dressing.nvim
- lazydev.nvim
- nvim-luapad

Tips for Plugin Authors

- **Respect User Configuration**: Provide sensible defaults, and allow customization
- **Mind the Performance**: Minimize blocking operations, use async APIs for blocking tasks
- **Document Everything**: all commands, configurations, and keybindings, automate vimdoc generation

- **Automated Testing**: Test critical behavior, use CI to automate testing
- **Share and Care**: Publish and maintain your plugin on GitHub and awesome-neovim

> ***Most Important: Have Fun!***

Thank You!

Let's build the Neovim ecosystem together!

Dive into plugin development, start by building and sharing your own plugin.

Feel free to reach out to me for any questions, ideas or feedback.

slides (https://github.com/2kabhishek/talks/blob/main/building-neovim-plugins.md)

> *for VimConf 2024, Akihabara, Tokyo, Japan*