Neovim Configuration

David Schmid

May 29, 2023

1 About Neovim

Neovim is a powerful, extensible, and highly customizable text editor built upon the foundations of Vim. This configuration file for Neovim ('init.lua') allows to tailor the editor to specific needs and preferences. You can unleash the power by customizing key mappings that resonate with your style, channeling the raw energy of plugins and sculpting the appearance. With each strum of configuration you transcend the ordinary editor to greatness.

By leveraging the Lua programming language, Neovim allows you to command the editor with unparalleled finesse. This manuscript ignites the flame of your Neovim journey, serving as a catalyst of boundless exploration. So go forth, fearless adventurer and may your Neovim journey be a symphony of joy and discovery.

2 Installation and Configuration

2.1 Installation

2.1.1 Installing Neovim

To install Neovim, visit the official Neovim GitHub page at
https://github.com/neovim/neovim/wiki/Installing-Neovim
and follow the instructions provided for your specific platform.

For Ubuntu users, it's advised against using the 'apt' package manager for installing Neovim as the supported version tends to lag behind. Instead, consider using the 'snap' package manager with the following command:

```
sudo snap install nvim --classic
```

This provides a more up-to-date version of Neovim. Alternatively, for the latest version, consider building Neovim from the source code. As of writing, the current version in use is 0.9.0.

2.1.2 Installing Dependencies

Before proceeding with the configuration, ensure that the required dependencies are installed on your system. These are detailed in Section 2.2.

2.1.3 Creating Configuration Directory

Create a directory in your system's configuration folder where the ‘init.lua‘ file will be placed:

```
mkdir -p ~/.config/nvim
```

2.1.4 Creating Symbolic Links

Create symbolic links from ‘ /.config/nvim/init.lua‘ and ‘ /.config/nvim/lua‘ pointing to the ‘ /dotfiles/nvim‘ directory:

```
ln -s ~/dotfiles/nvim/init.lua ~/.config/nvim/init.lua
ln -s ~/dotfiles/nvim/lua ~/.config/nvim/lua
```

2.1.5 Obtaining Configuration Files

Get the contents from the GitHub repository at
https://github.com/5n00py/dotfiles.

2.1.6 Launching Neovim

Start Neovim by running the ‘nvim‘ command in the terminal. Neovim will automatically load the ‘init.lua‘ file and apply the specified configurations.

2.1.7 Customizing Neovim

Open the ‘init.lua‘ file in Neovim and make any desired modifications to the settings, key mappings, and plugin configurations to tailor the editor to your preferences.

2.1.8 Installing Plugins

After launching Neovim with the custom ‘init.lua‘ file, execute the ‘:Lazy‘ command to trigger the installation and configuration of any lazily loaded plugins specified in this ‘init.lua‘ file. The command ensures that all necessary plugins are installed, and their respective configurations are applied.

2.1.9 Checking System Health

After launching Neovim with the custom ‘init.lua‘ file, run the ‘:checkhealth‘ command to analyze the setup and identify potential issues. Review the output, and resolve any detected problems according to the provided suggestions or by referring to the relevant plugin documentation.

2.2 Dependencies

This configuration for Neovim requires several external dependencies for optimal performance and extended functionality. Ensure that you have the following dependencies installed on your system:

- **Ripgrep:** A line-oriented search tool that recursively searches the current directory for a regex pattern. This is required for plugins like Telescope.
- **FD:** A simple, fast, and user-friendly alternative to the 'find' command. This is required for Telescope.
- **Build Essentials:** Basic development tools. At least 'make' should be installed for Telescope.
- **Clipboard Tool (xclip, pbcopy, etc.):** For clipboard support, a clipboard tool is required. 'xclip' is commonly used on Linux, while 'pbcopy' and 'pbpaste' are used on MacOS. Install via your package manager. For instance, 'sudo apt install xclip' can be used on Ubuntu.
- **Git:** Many Neovim plugins are hosted on GitHub, and Git is required to clone and update them.
- **Curl, Unzip, Gzip, Tar:** These utilities are often required for downloading and extracting files.
- **Plenary:** A Lua library used by several Neovim plugins for utility functions. It is installed as part of the plugin installation process.
- **LuaRocks:** The package manager for Lua modules. Some plugins may require Lua modules that can be installed using LuaRocks. Install via package manager. For instance, 'sudo apt install luarocks' can be used on Ubuntu.
- **Python Modules:** Neovim uses Python3 for some of its plugins and features. Specific requirements will be updated as necessary.
- **LSP-related Dependencies:** For the Language Server Protocol (LSP) to function, you need to have the respective language server installed for each programming language you plan to use in Neovim. These include:
  - Rust: Rust Analyzer
  - Python: Pyright
  - C: Clangd
  - JavaScript and TypeScript: Tsserver
  - Ruby: Solargraph
  - Go: Gopls
  - Shell: Bashls

  Each language server has its own installation instructions that may require some research.

2.3 Keybind Overview

This section provides an overview of the most commonly used key mappings in this Neovim configuration. This is not an exhaustive list. You can find a complete list of mappings by running the ':map' command or by pressing the <leader> key and <backspace> to show the Which-Key keybinds popup.

The Which-Key plugin provides a visual representation of key mappings. When you press <leader> followed by the <backspace> key, a popup window will appear, displaying a hierarchical menu of key mappings and their associated actions. As you start typing a key combination, Which-Key dynamically generates a popup window showing all available options and their associated actions, allowing you to navigate through the mappings with ease.

The <leader> key is set to the spacebar by default.

2.3.1 Tab Management

Key	Description	Mode
<tab>n	Open a new tab with a new empty buffer	Normal
<tab><tab>	Go to the next tab	Normal
<tab>p	Go to the previous tab	Normal
<tab>t	Open a new tab with a terminal in insert mode	Normal
<tab>c	Close the current tab	Normal

Tab navigation key mappings are associated with the <tab> key. In case the <tab> key is already associated with a different action in normal mode, you may need to adjust the key mappings to ensure smooth tab navigation within Neovim.

2.3.2 Window Management

Key	Description	Mode
<leader>w	Switch windows	Normal
<leader>q	Switch back to previously used window	Normal
<leader>/	Open empty window on the right	Normal
<leader>-	Open empty window below	Normal
<leader>J	Resize window 5 cells down	Normal
<leader>K	Resize window 5 cells up	Normal
<leader>H	Resize window 5 cells left	Normal
<leader>L	Resize window 5 cells right	Normal
<leader><Left>	Resize window 5 cells left	Normal
<leader><Right>	Resize window 5 cells right	Normal
<leader><Down>	Resize window 5 cells down	Normal
<leader><Up>	Resize window 5 cells up	Normal
<leader>h	Select window on the left	Normal
<leader>l	Select window on the right	Normal
<leader>j	Select window below	Normal
<leader>k	Select window above	Normal
<leader>e	Toggle NERDTree window on and off	Normal
<leader>ct	Change Transparency	Normal

2.3.3 Editing

Key	Description	Mode
<leader>a	Select all text in a buffer	Normal
<leader>d	Delete text without yanking	Visual
<leader>p	Paste text without yanking	Visual
<leader>tc	Toggle code comment	Normal / Visual
<leader>ta	Toggle autocompletion (cmp)	Normal / Visual
<leader>n	Edit nvim init.lua	Normal
<leader>u	Toggle Undotree	Normal

Neovim provides a wide range of standard editing functions and commands to manipulate and modify text. These include operations like copying, cutting, pasting, undoing, and redoing. You can access these functions through intuitive key combinations and commands, making it efficient to edit and manipulate text within Neovim. Familiarize yourself with these standard editing functions to enhance your editing experience in Neovim.

2.3.4 Telescope

Key	Description	Mode
<leader>ff	Find files	Normal
<leader>fg	Live grep	Normal
<leader>fb	List buffers	Normal
<leader>fh	Help tags	Normal
<leader>fw	Grep search on a WORD	Normal

2.3.5 LSP

Key	Description	Mode
gd	Go to definition	Normal
gD	Go to declaration	Normal
K	Show hover information	Normal
gr	Find references	Normal
<leader>fmt	Format code	Normal
<leader>d	Show diagnostics under cursor	Normal

2.3.6 Custom Commands

Format	Description
:Rep <c><w>	Repeat <c> times word(s) <w> and insert them to the buffer

3 References and External Resources

This section presents some external resources that can be useful in understanding and working with Neovim.

3.1 External Resources

- Neovim Documentation: https://neovim.io/doc/
- Lua Documentation: https://www.lua.org/docs.html
- Awesome Neovim: A curated collection of awesome plugins, themes, and resources for Neovim: https://github.com/rockerBOO/awesome-neovim
- Neovim GitHub Repository: https://github.com/neovim/neovim
- Neovim subreddit: https://www.reddit.com/r/neovim/

3.2 Manual Pages

Neovim provides a built-in help system which can be accessed using the ':help' command. Below are a few examples of how to use this system:

- `:help init.lua`: Provides an introduction to customizing.
- `:h key-notation`: Displays information about key notation.
- `:h :w`: Views the help page for the write command.