Neovim Guide: From Zero to Hero

Table of Contents

Introduction

This guide will help you learn how to use Neovim with this custom configuration. Neovim is a powerful, extensible text editor that builds upon Vim's philosophy of efficient text editing. With the right configuration, it becomes a full-featured IDE-like environment while maintaining its lightweight, keyboard-centric approach.

Getting Started

1. Install Neovim
2. Clone this config to `~/.config/nvim/`
3. Install Packer for plugin management:

```
git clone https://github.com/wbthomason/packer.nvim
~/.local/share/nvim/site/pack/packer/start/packer.nvim
```

4. Launch Neovim and run `:PackerSync` to install all plugins

Basic Vim Motions

Vim's power comes from its modal editing system and efficient movement commands.

Movement

Key	Action
`h`, `j`, `k`, `l`	Left, down, up, right
`w`	Move forward by word
`b`	Move backward by word
`e`	Move to end of word
`0`	Start of line

Key	Action
`$`	End of line
`gg`	Top of file
`G`	Bottom of file
`{`	Jump to previous paragraph
`}`	Jump to next paragraph
`Ctrl-d`	Move down half a page (centers cursor)
`Ctrl-u`	Move up half a page (centers cursor)
`%`	Jump to matching bracket
`fx`	Jump to next occurrence of character x
`tx`	Jump before next occurrence of character x

Editing

Key	Action
`i`	Insert mode before cursor
`a`	Insert mode after cursor
`o`	Insert new line below and enter insert mode
`O`	Insert new line above and enter insert mode
`x`	Delete character
`dw`	Delete word
`dd`	Delete line
`yy`	Yank (copy) line
`p`	Paste after cursor
`P`	Paste before cursor
`u`	Undo
`Ctrl-r`	Redo
`r`	Replace single character
`c`	Change (delete and enter insert mode)

Visual Mode

Key	Action

Key	Action
`v`	Visual mode (character selection)
`V`	Visual line mode
`Ctrl-v`	Visual block mode
Movement keys	Expand selection
`o`	Move to other end of selection

Search

Key	Action
`/pattern`	Search forward for pattern
`?pattern`	Search backward for pattern
`n`	Next search result (remapped to center cursor)
`N`	Previous search result (remapped to center cursor)
`*`	Search for word under cursor

Custom Keybindings

This configuration uses Space as the leader key.

File Navigation

Key	Action
`<leader>pv`	Open Netrw file explorer
`<leader>pf`	Find files with Telescope
`<C-p>`	Search Git files with Telescope
`<leader>ps`	Grep search in files

Harpoon (Quick File Navigation)

Key	Action
`<leader>a`	Add file to Harpoon
`<C-e>`	Toggle Harpoon quick menu
`<C-h>`	Navigate to Harpoon file 1
`<C-t>`	Navigate to Harpoon file 2
`<C-n>`	Navigate to Harpoon file 3

Key	Action
`<C-s>`	Navigate to Harpoon file 4

LSP (Language Server Protocol)

Key	Action
`gd`	Go to definition
`K`	Show hover information
`<leader>vws`	Workspace symbol search
`<leader>vd`	Show diagnostics in float
`[d`	Go to next diagnostic
`]d`	Go to previous diagnostic
`<leader>vca`	Code action
`<leader>vrr`	Show references
`<leader>vrn`	Rename symbol
`<C-h>` (insert mode)	Show signature help

Code Completion

Key	Action
`<C-p>`	Select previous item
`<C-n>`	Select next item
`<C-y>`	Confirm selection
`<C-Space>`	Open completion menu

Text Manipulation

Key	Action
`J` (visual mode)	Move selected lines down
`K` (visual mode)	Move selected lines up
`J` (normal mode)	Join lines and keep cursor position
`<leader>p` (visual mode)	Paste without yanking selection
`<leader>y`	Yank to system clipboard
`<leader>Y`	Yank line to system clipboard
`<leader>d`	Delete without yanking

Key	Action
`<leader>s`	Search and replace word under cursor
`<leader>f`	Format current buffer

Quickfix Navigation

Key	Action
`<C-k>`	Next quickfix item
`<C-j>`	Previous quickfix item
`<leader>k`	Next location list item
`<leader>j`	Previous location list item

Go Coding Snippets

Key	Action
`<leader>ee`	Insert error handling boilerplate
`<leader>ea`	Insert assertion error boilerplate
`<leader>ef`	Insert fatal error boilerplate
`<leader>el`	Insert logger error boilerplate

Misc

Key	Action
`<leader>x`	Make current file executable
`<C-c>`	Exit insert mode (alternative to Escape)
`<leader><leader>`	Source current file
`<C-f>`	Open tmux-sessionizer
`<leader>zig`	Restart LSP

Plugins Overview

This configuration includes several powerful plugins:

Telescope

Fuzzy finder for files, buffers, and more.

```
<leader>pf - Find files
<C-p> - Search git files
```

```
<leader>ps - Grep in files
```

Treesitter

Advanced syntax highlighting and code navigation.

Harpoon

Quick navigation between frequently used files.

```
<leader>a - Add file
<C-e> - Show menu
<C-h/t/n/s> - Jump to marked files
```

LSP & Autocompletion

Language server integration for smart code intelligence.

```
gd - Go to definition
K - Documentation
<leader>vca - Code actions
```

Undotree

Visualize file's undo history.

Fugitive

Git integration directly in Neovim.

Rose-pine

A clean, minimal color theme.

Common Workflows

Project Navigation

1. Open Neovim in project root
2. Use `<leader>pv` to browse files
3. Use `<leader>pf` or `<C-p>` to quickly find files
4. Mark important files with Harpoon (`<leader>a`)
5. Quickly switch between marked files with `<C-h/t/n/s>`

Coding with LSP

1. Navigate to definitions with `gd`

2. View documentation with `K`
3. Fix problems with `<leader>vca` (code actions)
4. Rename symbols with `<leader>vrn`
5. Use completion with `<C-n>` and `<C-p>`

Editing Text

1. Use visual mode (`v`) to select text
2. Move blocks with `J` and `K` in visual mode
3. Format code with `<leader>f`
4. Use `<leader>s` for quick search and replace
5. Use `=ap` to auto-indent paragraphs

Git Workflow

1. Use `:Git` commands for Git operations
2. Stage changes with `:Git add %`
3. Commit with `:Git commit`
4. Push/pull with `:Git push` and `:Git pull`

Tips and Tricks

Efficiency Boosters

- Learn to think in Vim motions (verb + noun)
- Use `.` to repeat last change
- Use macros (`q{register}` to record, `@{register}` to play)
- Combine commands like `ci"` (change inside quotes)

Configuration

- Core settings are in `lua/theabecaster/set.lua`
- Keymaps are in `lua/theabecaster/remap.lua`
- Plugin configurations are in `after/plugin/`

Customization

To customize this configuration:

1. Edit files in `lua/theabecaster/`
2. Add plugin configurations in `after/plugin/`
3. Install new plugins in `lua/theabecaster/packer.lua`
4. Run `:PackerSync` after changes

Happy coding with Neovim!