Neovim Lua-scriptable modal editor

CONCEPTS

Neovim keeps Vim's modal editing (see the vim sheet). This sheet covers what Neovim adds: Lua config, built-in LSP, terminal, health checks.

Config lives in ~/.config/nvim/init.lua (init.vim also works). Lua modules go in lua/ next to it and load with require('name').

Server binary installed separately
vim.lsp.config cmd, filetypes, roots
↓ defines
vim.lsp.enable starts on matching buffer
↓ attaches
Defaults + LspAttach gr* keys, K, diagnostics
↓ gives you
LSP in Neovim: no plugin needed for the client

Neovim ships only the LSP client. Language servers (lua-language-server, pyright, gopls...) are separate programs on your PATH.

DIFFERENCES FROM VIM

Y Yank to end of line

g › **c** › **c**
Toggle line comment

g › **c** › *<motion>*
Toggle comment over motion

g › **O**
Outline of symbols / headings

] › **d** / **[** › **d**
Next / previous diagnostic

] › **q** / **[** › **q**
Next / previous quickfix item

] › **b** / **[** › **b**
Next / previous buffer

***** / **#**
Search selected text (visual mode)

a › **n** / **i** › **n**
Grow visual selection by node

LUA CONFIG

:lua › *<code>*
Run a Lua expression

:= › *<expr>* Print a Lua value

:source › **%**
Reload the current file

:messages
Show earlier messages and errors

:set › **<opt>?**
Show an option's value

:Inspect
Highlight groups under cursor

:help › *<topic>*
Open help for a topic

:help › **lua-guide**
Lua in Neovim tutorial

Lua equivalents

Vimscript	Lua
`set number`	`vim.o.number = true`
`let g:x = 1`	`vim.g.x = 1`
`nnoremap a b`	`vim.keymap.set()`
`autocmd ...`	`nvim_create_autocmd()`
`command! ...`	`nvim_create_user_command()`

LSP

K Hover documentation

g › **r** › **n** Rename symbol

g › **r** › **a** Code action

g › **r** › **r** List references

g › **r** › **i**
Go to implementation

g › **r** › **t** Go to type definition

Ctrl + **]** Go to definition

Ctrl + **S**
Signature help (insert mode)

Ctrl + **X** › **Ctrl** + **O**
LSP completion (insert mode)

:lsp › **enable** › *<name>*
Start a configured server

:checkhealth › **vim.lsp**
Check LSP clients and logs

:lua › **vim.lsp.buf.format()**
Format the buffer

:lua › **vim.lsp.buf.definition()**
Go to definition

LSP (cont.)

The gr* keys exist only after a server attaches. Without one, g r just waits for the next key.

DIAGNOSTICS

Ctrl + **W** › **d**
Show diagnostic in float

TERMINAL

:terminal
Open a shell in a buffer

:terminal › *<cmd>*
Run a command in a terminal buffer

i / **a**
Enter terminal mode from normal

Ctrl + **** › **Ctrl** + **N**
Back to normal mode

Ctrl + **** › **Ctrl** + **O**
One normal command, then back

:split › **|** › **terminal**
Terminal in a split

:tnoremap › *<Esc>* › *<C-\><C-n>*
Make Esc leave terminal mode

In terminal mode every key goes to the program except Ctrl+\. Use :terminal instead of :! for interactive commands.

WINDOWS & BUFFERS

Ctrl + **W** › **s** / **Ctrl** + **W** › **v**
Split horizontally / vertically

Ctrl + **W** › **h** / **j** / **k** / **l**
Move between windows

Ctrl + **W** › **q** Close window

Ctrl + **W** › **o**
Keep only this window

PLUGINS & HEALTH

:checkhealth
Run all health checks

nvim › **--clean**
Start without config or plugins