

Minicurso Vim

André William Régis, Pedro Santi Binotto

UFSC

2024

Material de apoio

```
$ git clone --recursive  
https://github.com/AndreKuru/Minicurso-Vim.git
```

pratical-vim-examples

Vim



Principais modos

Normal: navegar e editar

Insert: inserção de texto

Visual: editar através de seleção

Command-line: retentor de usuários de primeira viagem

Digitação ideal



Imagen não adaptada.
Disponível em: Wikimedia

Movimentação básica

j : ↓

k : ↑

l : →

h : ←

Jogo para praticar:
vim-adventures.com

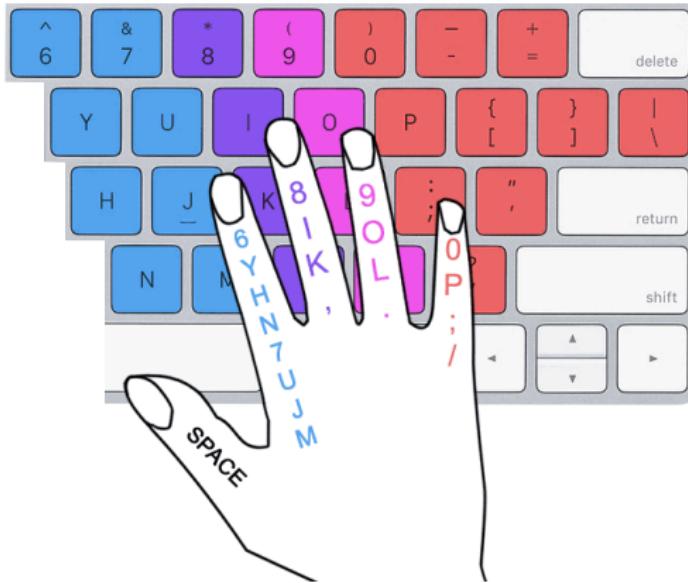


Imagen adaptada.
Disponible em: Wikimedia

Quando não se precisa manter a mão no mouse

u: undo change

dd: delete line

yy: yank line

p: put text



Mas se quiser
pode

:set mouse=a

Imagen não adaptada.
Disponível em:
Unsplash

Experimentando cada modo

i: switch to **I**nsert mode

v: switch to **V**isual mode

:: switch to Command-Line mode

Como sair do vim?

:q : quit

:q! : quit without writing

:w : write

:wq : write and quit

:x : write (if needed) and quit

Comando ponto

•: "Repeat last change [...]"

Mudanças comuns

x: delete [count] char(s)

d{motion} : delete

s: substitute [count] char(s)

c{motion} : change

[]: "[...]" are optional."

{}: "[...]" must appear, but which can take a number of different values."

[count] : "An optional number that may precede the command to multiply or iterate the command."

{motion} : "A command that moves the cursor."

Movimentação na linha

0: start of the line

\$: end of the line

^: start non-blank of the line

g_: end non-blank of the line

Movimentação através de palavras

w: word foward

word: "[...]

e: word end foward

letters, digits and underscores,
or a sequence of other non-blank
characters [...]"

b: word backward

blank characters: "space and tab"

ge: word end backward

Movimentação através de PALAVRAS

W: WORD foward

E: WORD end foward

B: WORD backward

gE: WORD end
backward

WORD: "[...]

sequence of non-blank characters
[...]"

blank characters: "space and tab"

Buscas dentro da linha

f: find char
to the right

t: till before char
to the right ;: "Repeat latest f, t, F or T [...]"

F: find char
to the left ,: "Repeat latest f, t, F or T
in opposite direction [...]"

T: till after char
to the left

Buscas no arquivo

/: search forward for
the pattern

?: search backward for
the pattern

*****: search forward for
the nearest word

#: search backward for
the nearest word

n: "Repeat latest "/" or "?" [...]"

N: "Repeat latest "/" or "?"
in opposite direction [...]"

Operadores

c: change

=: autoindent

d: delete

g~: swap case

y: yank

gu: make lowercase

>: shift right

gU: make uppercase

<: shift left

Teclas especiais

<Enter> : Enter

<CR> : Carriage Return

<Esc> : Escape

<Up> : Up arrow key

<S-...> : Shift + ...

<S-Down> : Shift + Down arrow key

<A-...> : Alt + ...

Ctrl-A : Ctrl + a

<C-...> : Ctrl + ...

<C-x> : Ctrl + x

Aritmética simples

<C-a> : add [count] to the number at or after the cursor

<C-x> : subtract [count] to the number at or after the cursor

Nem tudo se transforma

i: insert text

a: append text

I: insert text at the start of the line

A: append text at the end of the line

o: obtain new line
below

O: obtain new line
above

Correções rápidas

<C-h> : delete back one character (**<BS>**)

<C-w> : delete back one word

<C-u> : delete back to the start of line

Transições

<Esc> : switch to Normal mode

<C-[> : switch to Normal mode

<C-o> : switch to Insert Normal mode

Acessar registradores

<C-r>{register} : "Insert the contents of a register."

{register} :

+ : the clipboard contents

***** : the clipboard contents (X11: primary selection)

0 : contains the text from the most recent yank command

1-9 : contains the text deleted in ascending order by
the most recent delete or change command

= : the expression register

Inserir caracteres por código

<C-v>191 : £

<C-k>{char1}{char2}

<C-v>u00bf : £

<C-k>?I : £

<C-v>u00b0 : ũ

<C-k>12 : ¡

<C-v>{nondigit} : e.g.

<Home>

ga : print the ascii value

Mais um modo

R: Replace mode

gR: Virtual Replace mode

r: replace single character

gr: virtual replace single character

Cada modo uma orientação

<Esc> : switch to Normal mode

v : switch character-wise
Visual mode

<C-[> : switch to Normal mode

V : switch line-wise
Visual Line mode

gv : re-select last visual selection

<C-v> : switch block-wise
Visual Block mode

o : go to **other** end of selected text

Seleção entre delimitadores

a) or **ab**: a pair of (parentheses)

a]: a pair of [brackets]

a} or **aB**: a pair of {braces}

a>: a pair of <>

a': a pair of 'single quotes'

a`: a pair of ‘backticks’

a": a pair of ""double quotes"

at: a pair of tags

Seleção entre delimitadores

i) or **ib**: inside of (parentheses)

i]: inside of [brackets]

i} or **iB**: inside of {braces}

i>: inside of <>

i': inside of 'single quotes'

i“: inside of ‘backticks’

i": inside of "double quotes"

it: inside of tags

Seleção de objeto

aw : around of word

iw : inside of word

aW : around of WORD

iW : inside of WORD

as : around of sentence

is : inside of sentence

ap : around of paragraph

ip : inside of paragraph

Falando do ex

:print : print current line

:3print : print line 3

:2,4print : print lines 2 to 4

:<,>print : print lines selected from visual mode

Comandos básicos

:[range]print

[range] : {address}{,[{address}}]

:[range]join

{address} :

number : absolute line number

. : current line

\$: last line in the file

% : current file

/{pattern}[/] : next line [...]

?{pattern}[?] : previous line [...]

:[range]copy {address}

:[range]move {address}

Comandos básicos

:[range]delete [x]

[x] : "[into register x]"

a, b, c, d, e, f, g, h,

i, j, k, l, m, n, o, p,

q, r, s, t, u, v, w, x,

y, z

:[range]yank [x]

:[range]put [x]

Os mais populares

:[range]normal {address}

:[range]substitute/{pattern}/{string}/[flags]

:[range]global/{pattern}/[cmd]

[cmd] : command from command line

Repetição é a chave

@: : execute the last ex command

@@ : execute the last **@**

@{register} : execute the contents of register

q : record/stop macro

Comandos externos

:shell : start shell

:!{cmd} : execute **{cmd}** in the shell

Comandos externos integrados com o arquivo atual

:read !{cmd} : execute **{cmd}** in the shell
and put the output as text below cursor

:[range]write !{cmd} : execute **{cmd}** in the shell
with **[range]** lines as input

:[range] !{filter} : execute **{cmd}** in the shell
and put the output as text below cursor
with **[range]** lines as input

Rolar texto

<C-E> : scroll window
(**E**xtra lines)

<C-Y> : scroll window
(**Y**rineu)

<C-D> : scroll window
Downwards

<C-U> : scroll window
Upwards

<C-F> : scroll window
Forwards

<C-B> : scroll window
Backwards

Saltando pelo texto

[count]G : go to line number

% : jump to matching parenthesis

m{letter} : set **{mark}** **{letter}** at cursor position

`{mark} : jump to a **{mark}**

Navegando pelos saltos

<C-]> : "Jump to the definition of the keyword under the cursor."

<C-o> : go to older cursor position in jump list.

<C-i> : go to newer cursor position in jump list.

Autocomplete

<C-n> : find **n**ext match

<C-p> : find **p**revious match

Buffers, windows, tabs

Summary:

A buffer is the in-memory text of a file.

A window is a viewport on a buffer.

A tab page is a collection of windows[Moo].

Buffers no Vim

A window is a viewport onto a buffer. You can use multiple windows on one buffer, or several windows on different buffers.

A buffer is a file loaded into memory for editing. The original file remains unchanged until you write the buffer to the file[Moo].

Buffers no Vim

A buffer can be in one of three states:

active: The buffer is displayed in a window. If there is a file for this buffer, it has been read into the buffer. The buffer may have been modified since then and thus be different from the file.

hidden: The buffer is not displayed. If there is a file for this buffer, it has been read into the buffer. Otherwise it's the same as an active buffer, you just can't see it.

inactive: The buffer is not displayed and does not contain anything. Options for the buffer are remembered if the file was once loaded. It can contain marks from the **shada** file. But the buffer doesn't contain text[Moo].

Trabalhando com *buffers*

Buffer states			
state	displayed in window	loaded	:buffers shows
active	yes	yes	'a'
hidden	no	yes	'h'
inactive	no	no	,

Trabalhando com *buffers*

:e : edit file

:bn : buffer next

:bp : buffer previous

:bd : buffer delete

:b[n] : ex.: *buffer1 buffer2*

Windows: viewports para buffers

:split : Divide horizontalmente a tela com uma nova janela;

```
:split ./outro/arquivo.txt
```

:vsplit : Divide verticalmente a tela com uma nova janela;

```
:vsplit ./outro/arquivo.txt
```

:close[n] : Fecha uma janela (janela ativa ou **[n]**);

```
:close , :close 1
```

Windows: viewports para buffers

<C-W>s : **:split**

<C-W>v : **:vsplit**

Note: All **CTRL-W** commands can also be executed with **:wincmd**, for those places where a Normal mode command can't be used or is inconvenient[Moo].

Windows: viewports bara buffers

<C-W>[hjkI] : Move to window above/below/right of/left of current one.

<C-W>[HJKL] : Move window above/below/to the right/to the left.

Note: All **CTRL-W** commands can also be executed with **:wincmd**, for those places where a Normal mode command can't be used or is inconvenient[Moo].

Tab-pages

A tab page holds one or more windows. You can easily switch between tab pages, so that you have several collections of windows to work on different things[Moo].

Tab-pages

:tabe : tab edit

:tabn : tab next

:tabnew : tab new

:tabp : tab previous

:tabc : tab close

:tabs : tabs (plural de tab)

O sucessor moderno

Neovim is exactly what it claims to be. It fixes every issue I have with Vim.

—Geoff Greer

Full-screen Neovim looks cool as hell!

—DHH

A nice looking website, that's one thing Neovim did right.

—Bram Moolenaar



Imagen não adaptada.
Disponível em: [Neovim](#)

O sucessor moderno

API versionada, documentada;

Completamente configurável em Lua;

Ecossistema extensivo de plugins desenvolvidos pela comunidade;

Analisador sintático *TreeSitter* permite highlighting e navegação de código a nível de objetos sintáticos;

Cliente de LSP embutido (mesmo protocolo do *VSCode!*) habilita funcionalidades de inteligência de código (refatoração automática, *code actions*, navegação por referência, etc...);

Language Server Protocol

O que é LSP?

“Adding features like auto complete, go to definition, or documentation on hover for a programming language takes significant effort. Traditionally this work had to be repeated for each development tool, as each tool provides different APIs for implementing the same feature. A Language Server is meant to provide the language-specific smarts and communicate with development tools over a protocol that enables inter-process communication[mic].”

Language Server Protocol

Protocolo

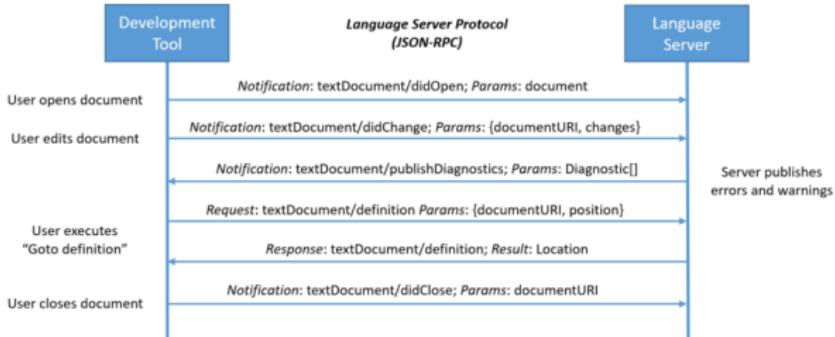


Imagen não adaptada.

Disponível em: Microsoft - Language Server Protocol

Language Server Protocol - Instalação e Configuração

Supporte nativo da API

```
2 lspconfig.rust_analyzer.setup {-
3   settings = {
4     ['pyright'] = {
5       python = {
6         analysis = {
7           diagnosticMode = "workspace",
8           typeCheckingMode = "off",
9         },
10        },
11      },
12      ['rust-analyzer'] = {},
13      ['gopls'] = {},
14      ['typescript-language-server'] = {},
15      ...
16    },

```

Além de oferecer *bindinds* nativas para a comunicação com LSPs, a configuração dos clientes pode ser feita de forma simples e direta através do pacote `nvim-lspconfig`

Language Server Protocol - Instalação e Configuração

`nvim-lspconfig + Mason.nvim = Configuração rápida, fácil, e portátil`



mason.nvim

Imagen não adaptada.
Disponível em: [Mason.nvim](#)

Portable package manager for Neovim that runs everywhere Neovim runs. Easily install and manage LSP servers, DAP servers, linters, and formatters.[mas]

Language Server Protocol - Instalação e Configuração

Um package-manager para servidores LSP

The screenshot shows a terminal window titled "mason.nvim" with the URL <https://github.com/williamboiman/mason.nvim>. The window displays a list of available LSP packages and their versions. The user is prompted to press 'g' for help or 'q' to quit. The list includes:

- (1) All (2) LSP (3) DAP (4) Linters (5) Formatter
- Language Filter: press <E-F> to apply filter
- Installed Press U to update 3 packages (lua-language-server, pylsight, ruff) (S)
- angular-language-server angularts
- bione
- clang
- css-lsp cssls
- eslint-lsp eslint
- html-lsp html
- json-lsp jsonls
- lua-language-server luau_ls ■ new version available: 3.10.3 → 3.10.5
- pylsight ■ new version available: 1.1.376 → 1.1.377
- ruff ■ new version available: 0.5.7 → 0.6.2
- solidity-ls solidity
- tailwindcss-language-server tailwindcss
- typescript-language-server tsserver
- typescript-tsparser typescript-tsparser
- Available (444)
 - actionlint
 - ada-language-server als
 - aiken
 - alex
 - ansible-language-server ansiblels
 - ansible-lint
 - antlr-language-server antlrsls
 - apex-language-server apex_ls
 - arduino-language-server arduino_language_server
 - asm-lsp asm_lsps
- luksas@luksas-MXe:~\$ mason
- [mason_outfile] --> selection primary: API versãoada, documentada; Completamente configurada.

At the bottom, the terminal shows the command "mason" and the status "NORMAL - [No Name] [-]" along with system information like "spaces: 2 utf-8 mason linux 1:1" and the timestamp "14:25:23-Aug-20".

Instalação de LSPs via *Mason.nvim*

TreeSitter

O motor sintático



“Tree-sitter is a parser generator tool and an incremental parsing library. It can build a concrete syntax tree for a source file and efficiently update the syntax tree as the source file is edited.”

TreeSitter

O motor sintático

General enough to parse any programming language;

Fast enough to parse on every keystroke in a text editor;

Robust enough to provide useful results even in the presence of syntax errors;

Dependency-free so that the runtime library (which is written in pure C) can be embedded in any application;

TreeSitter

O motor sintático

“The goal of nvim-treesitter is both to provide a simple and easy way to use the interface for tree-sitter in Neovim and to provide some basic functionality such as highlighting based on it.”

TreeSitter

O motor sintático

```

15 int ts_lua_add_language(lua_State *L)
16 {
17     if (!lua_gettop(L) < 2 || !lua_isstring(L, 1) || !lua_isstring(L, 2)) {
18         return luaL_error(L, "string expected");
19     }
20
21     const char *path = lua_tostring(L, 1);
22     const char *lang_name = lua_tostring(L, 2);
23
24     if (pmap_has(cstr_t)(langs, lang_name)) {
25         return 0;
26     }
27
28 #define BUFSIZE 128
29     char symbol_buf[BUFSIZE];
30     snprint(symbol_buf, BUFSIZE, "tree_sitter_ns", lang_name);
31     luaL_Buffer buf;
32     luaL_newbuffer(&buf, &symbol_buf);
33     luaL_pushstring(L, luaL_getbuffer(&buf));
34     return 0;
35 }
36
37 TSLanguage *ts_lang_parser(void);
38 if (uv_cldyn(&glib, symbol_buf, (void **)&lang_parser)) {
39     snprintf((char *)>IObuff, IOSIZE, "Failed to load parser: uv_dlopen: %s",
40             uv_dlerror(glib));
41     uv_dclose(glib);
42     luaL_pushstring(L, (char *)>IObuff);
43     return luaL_error(L);
44 }
45
46 TSLanguage *lang = lang_parser();
47 if (lang == NULL) {
48     return luaL_error(L, "Failed to load parser: internal error");
49 }
50
51 pmap_put(cstr_t)(langs, xstzdup(lang_name), lang);
52
53 lua_pushboolean(L, true);
54
55 return 1;
56 }
```



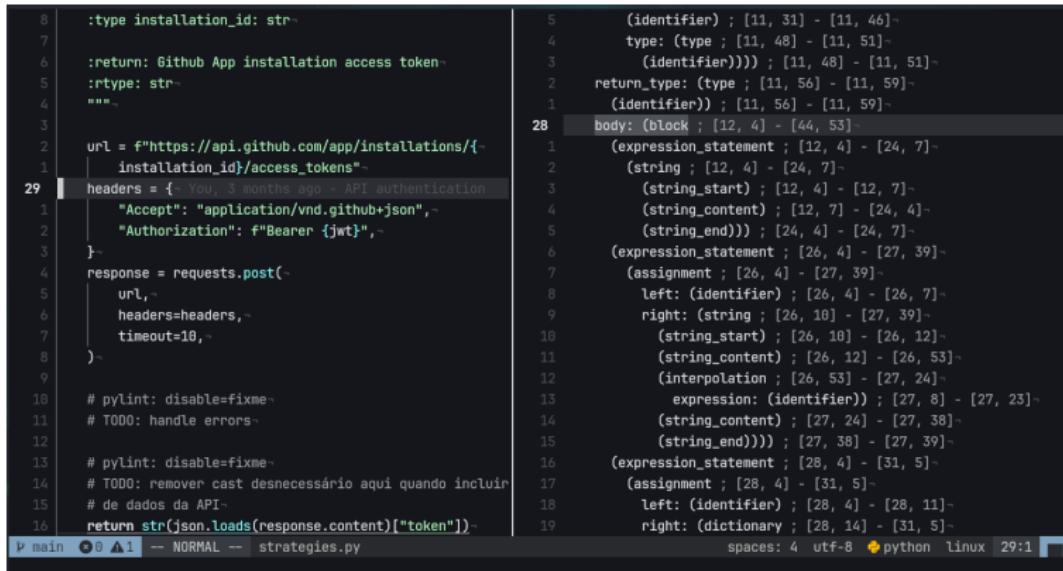
```

12 int ts_lua_add_language(lua_State *L)
13 {
14     if (!lua_gettop(L) < 2 || !lua_isstring(L, 1) || !lua_isstring(L, 2)) {
15         return luaL_error(L, "string expected");
16     }
17
18     const char *path = lua_tostring(L, 1);
19     const char *lang_name = lua_tostring(L, 2);
20
21     if (pmap_has(cstr_t)(langs, lang_name)) {
22         return 0;
23     }
24
25 #define BUFSIZE 128
26     char symbol_buf[BUFSIZE];
27     snprint(symbol_buf, BUFSIZE, "tree_sitter_ns", lang_name);
28     luaL_Buffer buf;
29     luaL_newbuffer(&buf, &symbol_buf);
30     luaL_pushstring(L, luaL_getbuffer(&buf));
31     return 0;
32 }
33
34 TSLanguage *lang = lang_parser();
35 if (uv_cldyn(&glib, symbol_buf, (void **)&lang_parser)) {
36     snprintf((char *)>IObuff, IOSIZE, "Failed to load parser: uv_dlopen: %s",
37             uv_dlerror(glib));
38     uv_dclose(glib);
39     luaL_pushstring(L, (char *)>IObuff);
40     return luaL_error(L);
41 }
42
43 pmap_put(cstr_t)(langs, xstzdup(lang_name), lang);
44
45 lua_pushboolean(L, true);
46
47 return 1;
48 }
```

Highlighting de sintaxe através da árvore construída

TreeSitter

O motor sintático



```

8   :type installation_id: str-
9
10  :return: Github App installation access token-
11  :rtype: str-
12  """
13
14  url = f"https://api.github.com/app/installations/{-
15  | installation_id}/access_tokens"-  

16  headers = {~ You, 3 months ago - API authentication
17      "Accept": "application/vnd.github+json",-
18      "Authorization": f"Bearer {jwt}"-
19  }-
20  response = requests.post(~
21      url,-
22      headers=headers,-
23      timeout=10,-
24  )-
25
26  # pylint: disable=fixme-
27  # TODO: handle errors-
28
29  # pylint: disable=fixme-
30  # TODO: remover cast desnecessário aqui quando incluir
31  # de dados da API-
32  return str(json.loads(response.content)[“token”])-

```

Parse tree structure (right side of the terminal):

```

5   (identifier) ; [11, 31] - [11, 46]-
6   type: (type ; [11, 48] - [11, 51]-
7       (identifier))) ; [11, 48] - [11, 51]-
8   return_type: (type ; [11, 56] - [11, 59]-
9       (identifier)) ; [11, 56] - [11, 59]-
10  body: (block ; [12, 4] - [44, 53]-
11      (expression_statement ; [12, 4] - [24, 7]-
12          (string ; [12, 4] - [24, 7]-
13              (string_start) ; [12, 4] - [12, 7]-
14              (string_content) ; [12, 7] - [24, 4]-
15              (string_end))) ; [24, 4] - [24, 7]-
16      (expression_statement ; [26, 4] - [27, 39]-
17          (assignment ; [26, 4] - [27, 39]-
18              left: (identifier) ; [26, 4] - [26, 7]-
19              right: (string ; [26, 10] - [27, 39]-
20                  (string_start) ; [26, 10] - [26, 12]-
21                  (string_content) ; [26, 12] - [26, 53]-
22                  (interpolation ; [26, 53] - [27, 24]-
23                      expression: (identifier)) ; [27, 8] - [27, 23]-
24                      (string_content) ; [27, 24] - [27, 38]-
25                      (string_end))) ; [27, 38] - [27, 39]-
26      (expression_statement ; [28, 4] - [31, 5]-
27          (assignment ; [28, 4] - [31, 5]-
28              left: (identifier) ; [28, 4] - [28, 11]-
29              right: (dictionary ; [28, 14] - [31, 5]-

```

Terminal status bar:

- main
- 0 1 1 -- NORMAL -- strategies.py
- spaces: 4 utf-8 ♦ python linux 29:1

A ferramenta permite inspecionar a estrutura da árvore sintática em tempo real

Plugins

Plugin managers

pathogen

vundle

vim-plug

packer.nvim

lazy.nvim

Plugins

Plugin managers

The screenshot shows the user interface of the `lazy.nvim` plugin manager. At the top, there's a navigation bar with links for `Install (I)`, `Update (U)`, `Sync (S)`, `Clean (X)`, `Check (C)`, `Log (L)`, `Restore (R)`, `Profile (P)`, `Debug (D)`, and `Help (?)`. Below the bar, it says `Total: 33 plugins`.

The main area is divided into two sections: `Log (6)` and `Loaded (30)`.

- Log (6):**
 - `mason.nvim` (marked as having updates available) with commit details: `alaf301 chore: update generated code (#767) (4 hours ago)`
 - `neogit` (marked as having updates available) with commits: `Od6002c feat: commit --fixup and --squash (4 days ago)`, `ee6f6ea refactor: rebase and commit selection (4 days ago)`, `2444769 fixup! fix: use parse_log for rebase commit view (4 days ago)`, `d12acd0 chore: ignore redefined-local (4 days ago)`
 - `noice.nvim` (marked as having updates available) with commit: `7dac8ce chore(build): auto-generate vimdoc (2 hours ago)`
 - `nvim-treesitter-context` (marked as having updates available) with commit: `d28654b ci: bump action versions (34 minutes ago)`
 - `nvim-treesitter-textobjects` (marked as having updates available) with commit: `e2ee8fd ci: bump stylua version (58 minutes ago)`
 - `playground` (marked as having updates available) with commit: `3421bbb ci: bump action versions (52 minutes ago)`
- Loaded (30):**
 - `animation.nvim` (`0.00ms`)
 - `windows.nvim`
 - `dressing.nvim` (`0.39ms`)
 - `VeryLazy`
 - `drop.nvim` (`0.48ms`)
 - `VimEnter`
 - `firenvim` (`0.16ms`)
 - `start`
 - `flit.nvim` (`0.02ms`)
 - `leap.nvim`
 - `github-notifications.nvim` (`0.04ms`)
 - `lualine.nvim`
 - `gruvbox.nvim` (`0.05ms`)
 - `start`
 - `lazy.nvim` (`4.7ms`)
 - `init.lua`
 - `lean-ast.nvim` (`0.02ms`)
 - `lean.nvim`

Instalação interativa de plugins com `lazy.nvim`

Lua

Brasil mentioned!?!?!



Imagen não adaptada.
Disponível em: www.lua.org

designed and developed at

PUC
RIO

Lua

API

The "Vim API" inherited from Vim: Ex-commands and builtin-functions as well as user-functions in Vimscript. These are accessed through `vim.cmd()` and `vim.fn` respectively.

The "Nvim API" written in C for use in remote plugins and GUIs; see `api`. These functions are accessed through `vim.api`.

Recomendações

vim-adventures.com

```
$ vimtutor
```

```
:help
```

Referência

-  *Mason.nvim - portable package manager for neovim that runs everywhere neovim runs,*
<https://github.com/williamboman/mason.nvim>,
Accessed: 2024-09-1.
-  *Microsoft - language server protocol*, <https://microsoft.github.io/language-server-protocol/>,
Accessed: 2024-09-1.
-  B Moolenaar, *Vim reference manual*, Accessed: 2024-09-1.
-  Drew Neil, *Practical vim: Edit text at the speed of thought*,
Practical Vim (2015), 1–356.