NvChad (English version)

A book from the Documentation Team

Version : 2024/03/20

Rocky Documentation Team

Table of contents

1. Licence 7

2. Introduction 8

2.1 Using Neovim as an IDE 9

2.1.1 Main Features 10

2.2 References 12

2.2.1 Lua 12

2.2.2 Neovim 13

2.2.3 LSP 14

2.2.4 tree-sitter 14

3. Additional Software Needed 15

3.1 RipGrep 15

3.2 RipGrep Verification 16

3.3 Lazygit 17

4. Install Neovim 19
4.1 Introduction to Neovim 19
4.1.1 Community of developers 19
4.1.2 Key Features 19
4.2 Neovim Installation 21
4.2.1 Uninstall 22
4.3 Neovim Basic 0
5. Turning Neovim into an advanced IDE 0
5.1 Pre-requisites 0
5.1.1 Preliminary Operations 0

5.2 Installation 0
5.2.1 Clone configuration 0
5.2.2 Bootstrap 0
5.3 Configuration Structure 0
5.3.1 Basic structure 0
5.3.2 Template Structure 0
5.4 Structure analysis 0
5.5 Main keyboard keys 0
6. Example configuration 0
6.1 Introduction 0
6.2 Installation 0
6.3 Structure 0
6.4 Structure Analysis 0
6.4.1 Main Files 0
6.4.2 Configs folder 0
6.5 Conclusion 0
7. Nerd Fonts - Fonts for Developers 0
7.1 Download 0
7.2 Installation 0
7.3 Configuration 0
8. `vale` in NvChad (Neovim) 0
8.1 Introduction 0
8.2 Prerequisites 0
8.2.1 Installation of nvim-lint 0
8.3 Installing `vale` using Mason 0
8.3.1 Configuring and initializing `vale` 0
8.3.2 The `lint.lua` file changes 0
8.4 Conclusions and final thoughts 0

9. Marksman - code assistant 0
9.1 Objectives 0
9.2 Requirements and skills 0
9.3 Installation of Marksman 0
9.4 Integration into the editor 0
9.5 Use of marksman 0
9.6 Main keys 0
9.7 Conclusion 0
10. Basic configuration plugins 0
10.1 Main Plugins 0
10.2 LSP functionality 0
10.3 Lua Code 0
10.4 Mixed Plugins 0
10.5 File Management 0
11. Plugins Manager 0
11.1 Main Features 0
11.2 Preliminary Operations 0
11.3 Inserting a Plugin 0
11.4 Removing a plugin 0
11.5 Updating Plugins 0
11.6 Additional Features 0
11.7 Synchronization 0
12. NvChad Interface 0
12.1 Tabufline 0
12.2 Middle Section - Open Buffers 0
12.3 Statusline 0
12.4 Integrated Help 0
12.5 NvimTree 0
13. Editing with NvChad 0
13.1 Open a file 0
13.2 Working with the Editor 0
13.2.1 Text Selection 0
13.2.2 Text search 0
13.3 Saving the Document 0
14. NvimTree - File Explorer 0
14.1 Working with the File Explorer 0
14.1.1 Select a File 0
14.1.2 Opening a File 0

14.1.3 File Management 0

14.2 Advanced Features 0

14.3 Conclusion 0

15. Markdown Preview 0

15.1 Introduction 0

15.1.1 Peek.nvim 0

15.1.2 Markdown-preview.nvim 0

15.2 Conclusions and final thoughts 0

16. Project Manager 0

16.1 Introduction 0

16.1.1 Plugin installation 0

16.1.2 Using the Project Manager 0

16.1.3 Additional functions 0

16.1.4 Mapping 0

16.2 Conclusions and final thoughts 0

1. Licence

The documents and their sources are freely downloadable from:

- https://docs.rockylinux.org
- https://github.com/rocky-linux/documentation

Our media sources are hosted at github.com. You'll find the source code repository where the version of this document was created.

From these sources, you can generate your own personalized training material using mkdocs. You will find instructions for generating your document here.

How can I contribute to the documentation project?

You'll find all the information you need to join us on our git project home page.

We wish you all a pleasant reading and hope you enjoy the content.

2. Introduction

Release 2.5 changes

With the release of version 2.5, the editor's developers significantly changed the configuration's structure. The most significant changes concern the following aspects:

- The configuration transformation into a Neovim plugin can then be updated using the *lazy.nvim* plugins manager
- Removing the `custom` folder for editor customization (now integrated into the main folder). A migration script is provided for current users.
- The mappings have been changed and no longer use nvchad's custom syntax, instead nvim's **vim.keymap.set** is used.

As a result, some pages of the guide, especially the whole part concerning the installation of NvChad and the subsequent installation of plugins, appear incorrect. The guide is **under revision** and will be updated soon.

Throughout this book, you will find ways to implement Neovim, along with NvChad, to create a fully functional **I**ntegrated **D**evelopment **E**nvironment (IDE).

I say "ways" because there are many possibilities. The author focuses here on using these tools for writing markdown, but if markdown isn't your focus, don't worry simply read on. If you are unfamiliar with either of these tools (NvChad or Neovim), then this book will give you an introduction to both, and if you step through these documents, you'll soon realize that you can set up this environment to be a huge help for whatever your programming or script writing needs are.

Want an IDE that will help in writing Ansible playbooks? You can get that! Want an IDE for Golang? That's available too. Simply want a good interface for writing BASH scripts? It's also available.

Through the use of **L**anguage **S**erver **P**rotocols and linters, you can setup an environment that is customized just for you. The best part is that once you have the environment setup, it can quickly be updated when new changes are available through the use of lazy.nvim and Mason, both of which are covered here.

Because Neovim is a fork of Vim, the overall interface will be familiar to *vim* users. If you aren't a *vim* user, you will pick up on the syntax of the commands quickly using this book. There's a lot of information covered here but it's easy to follow along, and once you've completed the content, you'll know enough to build your own IDE for *your* needs with these tools.

It was the author's intent **not** to break this book down into chapters. The reason is that this implies an order that must be followed and, for the most part, that's not necessary. You *will* want to start with this page, read and follow the "Additional Software", "Install Neovim" and "Install NvChad" sections, but from there, you can choose how you want to proceed.

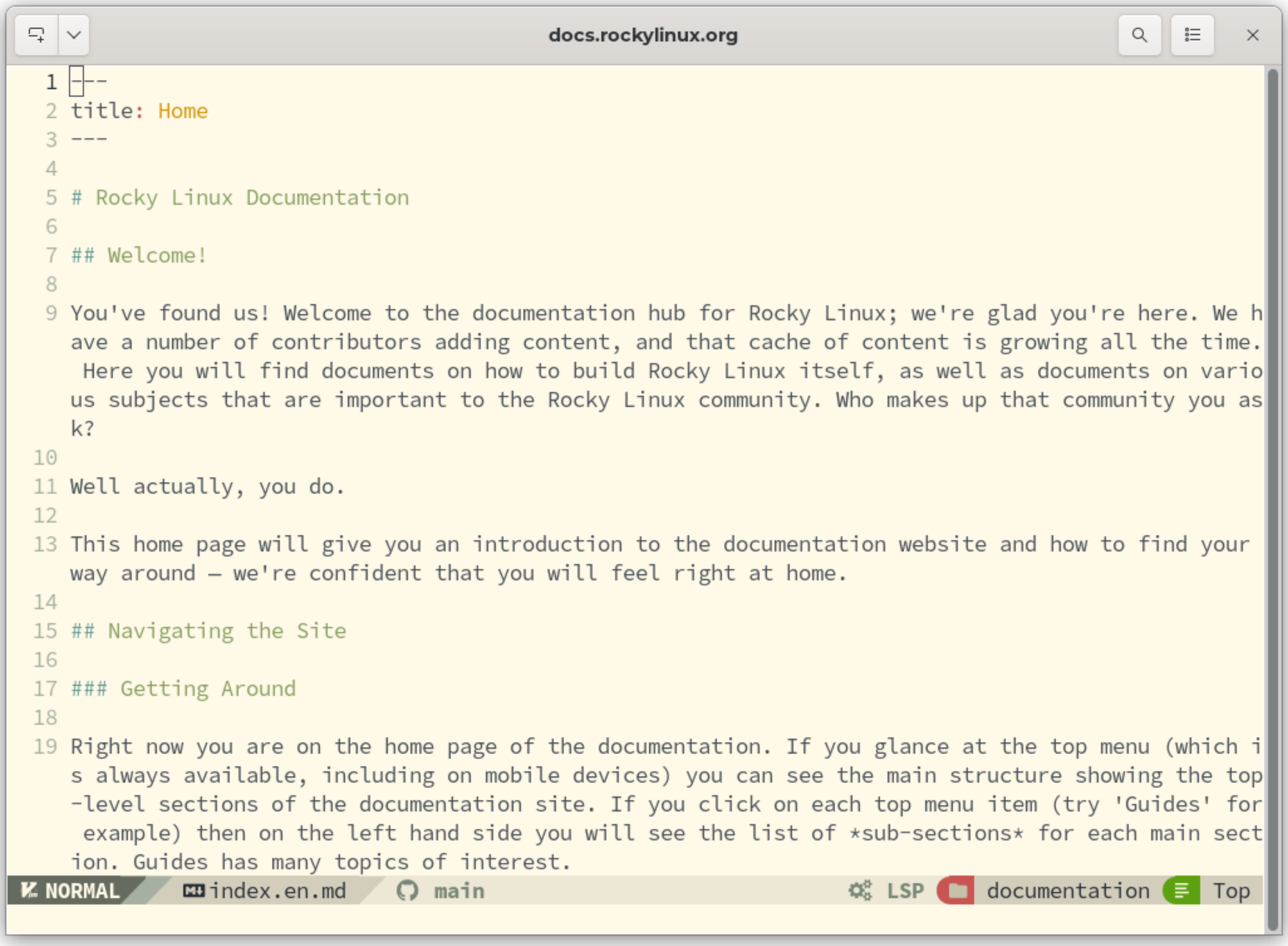
2.1 Using Neovim as an IDE

The basic installation of Neovim provides an excellent editor for development, but it cannot yet be called an IDE; all the more advanced IDE features, even if already preset, are not yet activated. To do this we need to pass the necessary configurations to Neovim, and this is where NvChad comes to our aid. This allows us to have a basic configuration out of the box with just one command!

The configuration is written in Lua, a very fast programming language that allows NvChad to have startup and execution times for commands and keystrokes that are very fast. This is also made possible by the Lazy loading technique used for plugins that loads them only when required.

The interface turns out to be very clean and pleasant.

As the developers of NvChad are keen to point out, the project is only intended to be a base on which to build your own personal IDE. Subsequent customization is done through the use of plugins.

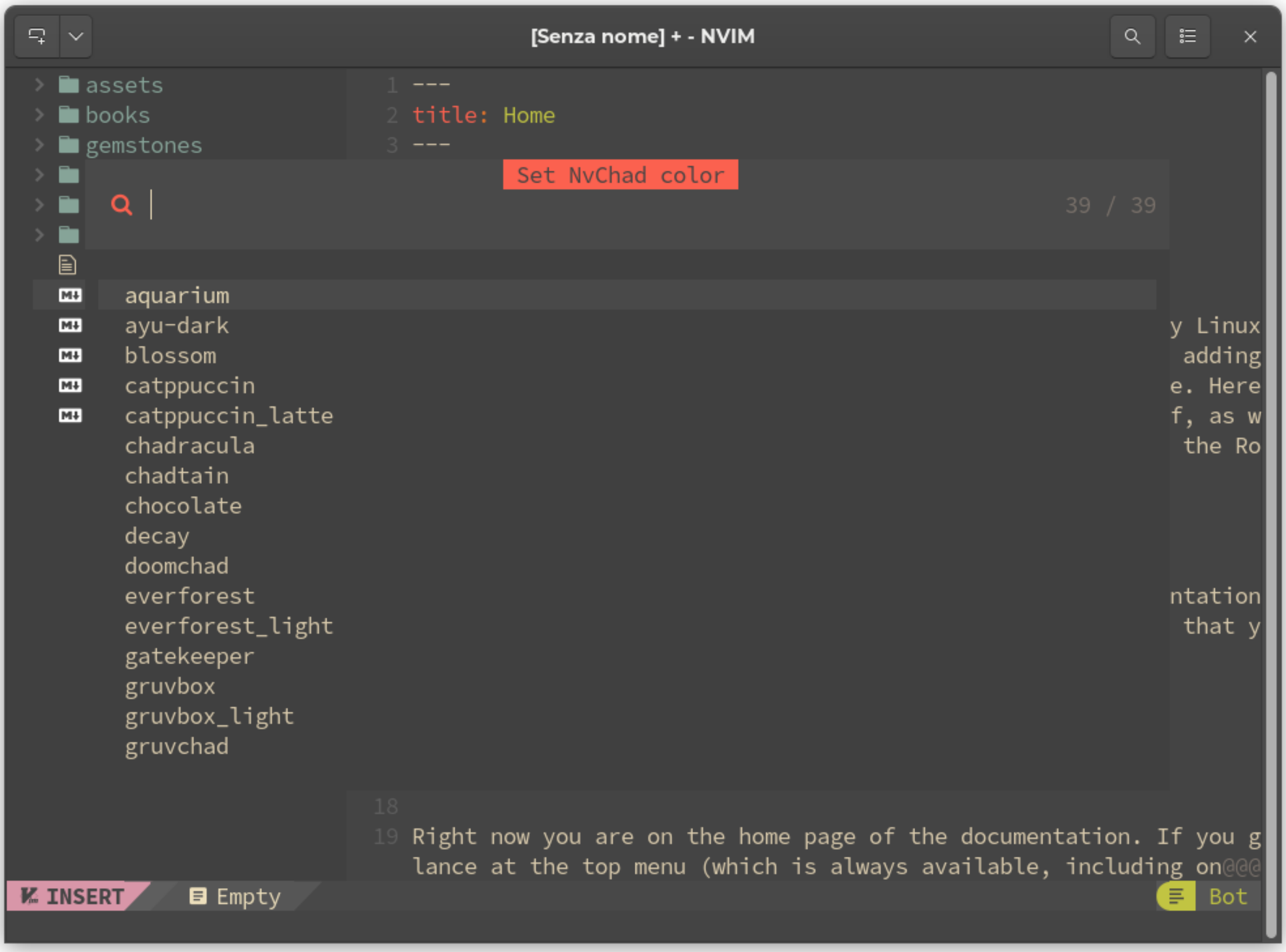


```
---
title: Home
---

# Rocky Linux Documentation

## Welcome!

You've found us! Welcome to the documentation hub for Rocky Linux; we're glad you're here. We have a number of contributors adding content, and that cache of content is growing all the time. Here you will find documents on how to build Rocky Linux itself, as well as documents on various subjects that are important to the Rocky Linux community. Who makes up that community you ask?

Well actually, you do.

This home page will give you an introduction to the documentation website and how to find your way around — we're confident that you will feel right at home.

## Navigating the Site

### Getting Around

Right now you are on the home page of the documentation. If you glance at the top menu (which is always available, including on mobile devices) you can see the main structure showing the top-level sections of the documentation site. If you click on each top menu item (try 'Guides' for example) then on the left hand side you will see the list of *sub-sections* for each main section. Guides has many topics of interest.
```

2.1.1 Main Features

- **Designed to be fast.** From the choice of programming language to techniques for loading components, everything is designed to minimize execution time.
- **Attractive Interface.** Despite being a *cli* application the interface looks modern and beautiful graphically, plus all the components fit the UI perfectly.
- **Extremely Configurable.** Due to the modularity derived from the base application (NeoVim), the editor can be adapted perfectly to one's needs. However, remember that when we talk about customization, we are referring to functionality, and not to the appearance of the interface.
- **Automatic update mechanism.** The editor comes with a mechanism (through the use of *git*) that allows updates with a simple `:NvChadUpdate` command.

- **Powered by Lua.** NvChad's configuration is written entirely in *lua*, which allows it to integrate seamlessly into Neovim's configuration by taking advantage of the full potential of the editor on which it is based.
- **Numerous inbuilt themes.** The configuration already includes a large number of themes to use, always keeping in mind that we are talking about a *cli* application, themes can be selected with the `<leader> + th` key.



```
---
title: Home
---
```

```
aquarium
ayu-dark
blossom
catppuccin
catppuccin_latte
chadracula
chadtain
chocolate
decay
doomchad
everforest
everforest_light
gatekeeper
gruvbox
gruvbox_light
gruvchad
```

```
Right now you are on the home page of the documentation. If you g
lance at the top menu (which is always available, including on@@@
```

2.2 References

2.2.1 Lua

What is Lua?

Lua is a robust, lightweight, scripting language that supports a variety of programming methods. The name "Lua" comes from the Portuguese word meaning "moon."

Lua was developed at the Catholic University of Rio de Janeiro by Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. The development was necessary for them because until 1992 Brazil was subject to strict import regulations for hardware and software, so out of sheer necessity, these three programmers developed their own scripting language called Lua.

Because Lua focuses primarily on scripts, it is rarely used as a stand-alone programming language. Instead, it is most often used as a scripting language that can be integrated (embedded) into other programs.

Lua is used in the development of video games and game engines (Roblox, Warframe..), as a programming language in many network programs (Nmap, ModSecurity..), and as a programming language in industrial programs. Lua is also used as a library that developers can integrate into their programs to enable scripting functionality by acting solely as an integral part of the host application.

How Lua works

There are two main components of Lua:

- The Lua interpreter
- The Lua virtual machine (VM)

Lua is not interpreted directly through a Lua file like other languages, for example Python. Instead, it uses the Lua interpreter to compile a Lua file into bytecode. The Lua interpreter is highly portable and capable of running on a multitude of devices.

Key Features

- Speed: Lua is considered one of the fastest programming languages among interpreted scripting languages; it can perform very performance-heavy tasks faster than most other programming languages.
- Size: Lua is tiny compared to other programming languages. This small size is ideal for integrating Lua into multiple platforms, from embedded devices to game engines.
- Portability and integration: Lua's portability is almost unlimited. Any platform that supports the standard C compiler can run Lua without problems. Lua does not require complex rewrites to be compatible with other programming languages.
- Simplicity: Lua has a simple design but provides powerful functionality. One of the main features of Lua is meta-mechanisms, which allow developers to implement their own functionality. The syntax is simple and easily understood, so that anyone can learn Lua and use it in their own programs.
- License: Lua is free and open-source software distributed under the MIT license. This allows anyone to use it for any purpose without paying any license or royalties.

2.2.2 Neovim

Neovim is described in detail on its dedicated page so we will just dwell on the main features, which are:

- Performance: Very fast.
- Customizable: Wide ecosystem of plugins and themes.
- Syntax highlighting: Integration with Treesitter and LSP, (requires some additional configurations).
- Multiplatform: Linux, Windows and macOS
- License: Mit: A short and simple permissive license with conditions requiring only the preservation of copyright and license notices.

2.2.3 LSP

What is the **L**anguage **S**erver **P**rotocol?

A language server is a standardized language library that uses its own procedure (protocol) to provide support for functions such as autocomplete, goto definition, or mouseover definitions.

The idea behind the Language Server Protocol (LSP) is to standardize the communication protocol between tools and servers, so that a single language server can be reused in multiple development tools. In this way, developers can simply integrate these libraries into their editors and reference existing language infrastructures, instead of customizing their code to include them.

2.2.4 tree-sitter

Tree-sitter basically consists of two components: a parser generator, and an incremental parsing library. It can build a syntactic tree of the source file and efficiently update it with each change.

A parser is a component that decomposes data into smaller elements to facilitate its translation into another language, or as in our case, to be then passed to the parsing library. Once the source file has been decomposed, the parsing library parses the code and transforms it into a syntactic tree, allowing the structure of the code to be manipulated more intelligently. This makes it possible to improve (and speed up)

- syntax highlighting
- code navigation
- refactoring
- text objects and movements

LS and tree-sitter complementarity.

Although it may seem that the two services (LSP and tree-sitter) are redundant, they are actually complementary in that LSP works at the project level while tree-sitter works only on the open source file.

Now that we have explained a bit about the technologies used to create the IDE we can move on to the Additional Software needed to configure our NvChad.

3. Additional Software Needed

There are several pieces of additional software that, while not required, will aid in the overall use of NvChad. The sections below will walk you through that software and its uses.

3.1 RipGrep

`ripgrep` is a line-oriented search tool that recursively searches the current directory for a *regex* (regular expression) pattern. By default, *ripgrep* respects the rules of *gitignore* and automatically skips hidden files/directories and binaries.

Ripgrep offers excellent support on Windows, macOS and Linux, with binaries available for each release.

Install RipGrep from EPEL

In both Rocky Linux 8 and 9, you can install RipGrep from the EPEL. To do this, install the `epel-release`, upgrade the system, and then install `ripgrep`:

```
sudo dnf install -y epel-release
sudo dnf upgrade
sudo dnf install ripgrep
```

Install RipGrep using cargo

Ripgrep is software written in *Rust* and is installable with the `cargo` utility. Note, however, that `cargo` is not installed by the default installation of *rust*, so you have to install it explicitly. If you run into errors using this method, revert back to installing from the EPEL.

```
dnf install rust cargo
```

Once the necessary software is installed, we can install `ripgrep` with:

```
cargo install ripgrep
```

The installation will save the `rg` executable in the `~/.cargo/bin` folder which is outside the PATH, to use it at the user level we will link it to `~/.local/bin/`.

```
ln -s ~/.cargo/bin/rg ~/.local/bin/
```

3.2 RipGrep Verification

At this point we can check that everything is okay with:

```
rg --version
ripgrep 13.0.0
-SIMD -AVX (compiled)
+SIMD +AVX (runtime)
```

RipGrep is needed for recursive searches with `:Telescope`.

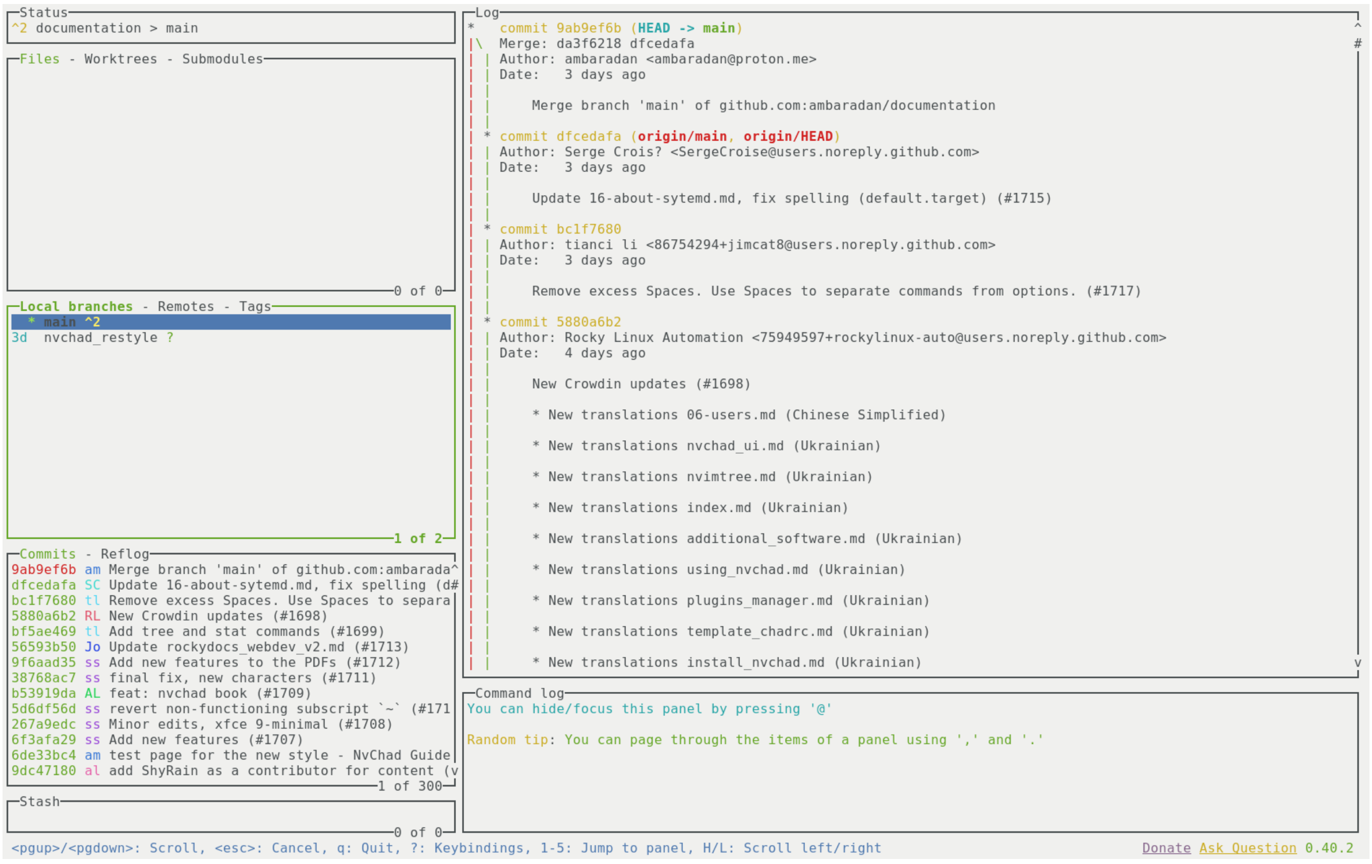
3.3 Lazygit

LazyGit is an ncurses-style interface that allows you to perform all `git` operations in a more user-friendly way. It is required by the lazygit.nvim plugin. This plugin makes it possible to use LazyGit directly from NvChad, it opens a floating window from where you can perform all operations on your repositories, thus allowing you to make all changes to the *git repository* without leaving the editor.

To install it we can use the repository for Fedora. On Rocky Linux 9 it works perfectly.

```
sudo dnf copr enable atim/lazygit -y
sudo dnf install lazygit
```

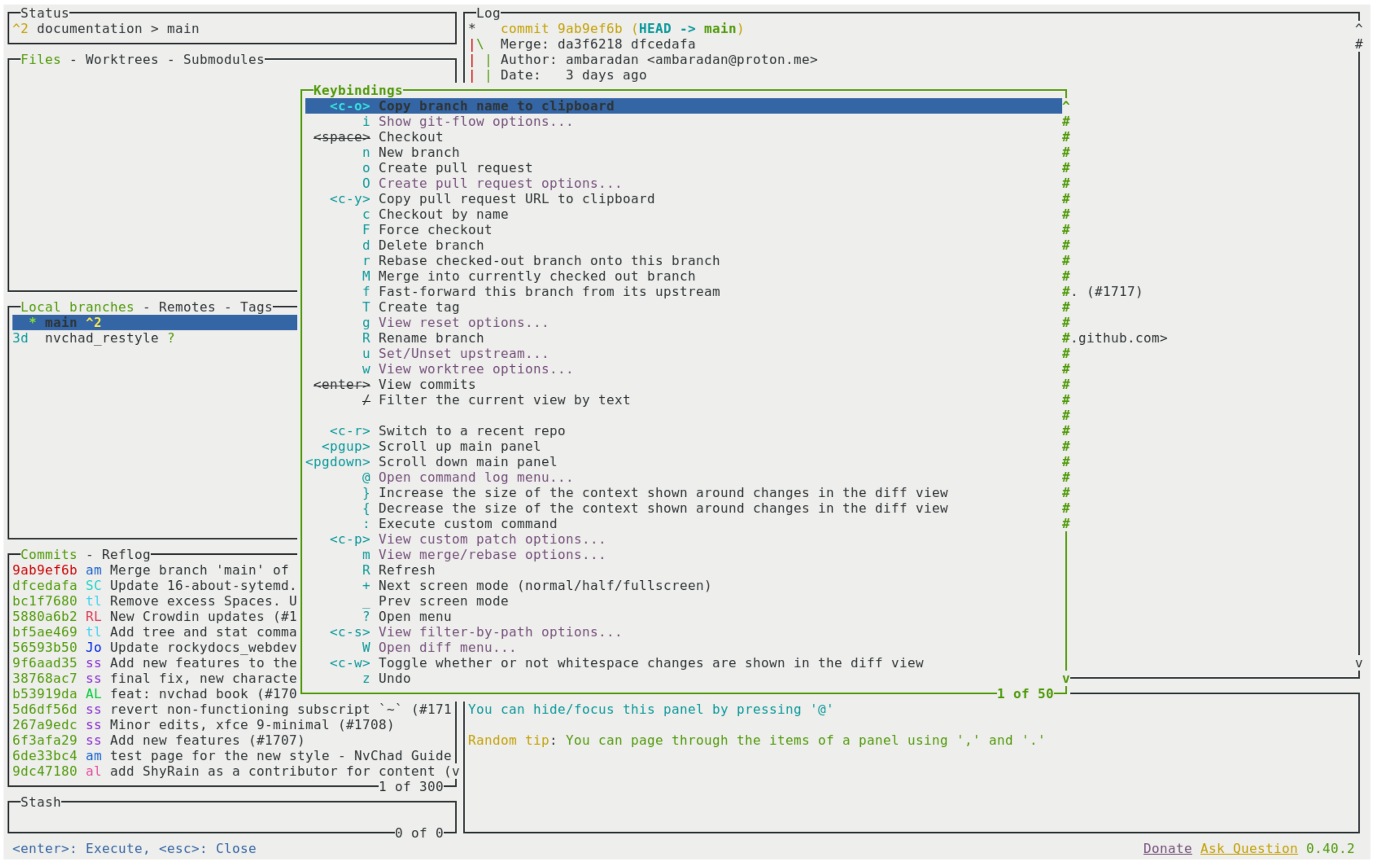
Once installed we open a terminal and type the command `lazygit` and an interface similar to this will appear:



```
Status
^2 documentation > main

Files - Worktrees - Submodules
                                                  0 of 0

Local branches - Remotes - Tags
  * main ^2
3d  nvchad_restyle ?
                                                  1 of 2

Commits - Reflog
9ab9ef6b am Merge branch 'main' of github.com:ambarada^
dfcedafa SC Update 16-about-sytemd.md, fix spelling (d#
bc1f7680 tl Remove excess Spaces. Use Spaces to separa
5880a6b2 RL New Crowdin updates (#1698)
bf5ae469 tl Add tree and stat commands (#1699)
56593b50 Jo Update rockydocs_webdev_v2.md (#1713)
9f6aad35 ss Add new features to the PDFs (#1712)
38768ac7 ss final fix, new characters (#1711)
b53919da AL feat: nvchad book (#1709)
5d6df56d ss revert non-functioning subscript `~` (#171
267a9edc ss Minor edits, xfce 9-minimal (#1708)
6f3afa29 ss Add new features (#1707)
6de33bc4 am test page for the new style - NvChad Guide
9dc47180 al add ShyRain as a contributor for content (v
                                                1 of 300

Stash
                                                  0 of 0
```

```
Log
*   commit 9ab9ef6b (HEAD -> main)
|\  Merge: da3f6218 dfcedafa
| | Author: ambaradan <ambaradan@proton.me>
| | Date:   3 days ago
| |
| |     Merge branch 'main' of github.com:ambaradan/documentation
| |
| * commit dfcedafa (origin/main, origin/HEAD)
| | Author: Serge Crois? <SergeCroise@users.noreply.github.com>
| | Date:   3 days ago
| |
| |     Update 16-about-sytemd.md, fix spelling (default.target) (#1715)
| |
| * commit bc1f7680
| | Author: tianci li <86754294+jimcat8@users.noreply.github.com>
| | Date:   3 days ago
| |
| |     Remove excess Spaces. Use Spaces to separate commands from options. (#1717)
| |
| * commit 5880a6b2
| | Author: Rocky Linux Automation <75949597+rockylinux-auto@users.noreply.github.com>
| | Date:   4 days ago
| |
| |     New Crowdin updates (#1698)
| |
| |     * New translations 06-users.md (Chinese Simplified)
| |
| |     * New translations nvchad_ui.md (Ukrainian)
| |
| |     * New translations nvimtree.md (Ukrainian)
| |
| |     * New translations index.md (Ukrainian)
| |
| |     * New translations additional_software.md (Ukrainian)
| |
| |     * New translations using_nvchad.md (Ukrainian)
| |
| |     * New translations plugins_manager.md (Ukrainian)
| |
| |     * New translations template_chadrc.md (Ukrainian)
| |
| |     * New translations install_nvchad.md (Ukrainian)

Command log
You can hide/focus this panel by pressing '@'

Random tip: You can page through the items of a panel using ',' and '.'

<pgup>/<pgdown>: Scroll, <esc>: Cancel, q: Quit, ?: Keybindings, 1-5: Jump to panel, H/L: Scroll left/right     Donate Ask Question 0.40.2
```

With the `?` key, we can bring up the menu with all available commands.



```
┌Status─────────────────────────────────────────┐ ┌Log──────────────────────────────────────────────────────────────────────────────────────┐
│^2 documentation > main                        │ │*   commit 9ab9ef6b (HEAD -> main)                                                      ^
└───────────────────────────────────────────────┘ ││\  Merge: da3f6218 dfcedafa                                                            #
┌Files - Worktrees - Submodules─────────────────┐ ││ | Author: ambaradan <ambaradan@proton.me>                                             │
│                                               │ ││ | Date:   3 days ago                                                                  │
│                                  ┌Keybindings──────────────────────────────────────────────────────────────────────────────┐
│                                  │   <c-o> Copy branch name to clipboard                                                   ^
│                                  │       i Show git-flow options...                                                        #
│                                  │ <space> Checkout                                                                        #
│                                  │       n New branch                                                                      #
│                                  │       o Create pull request                                                             #
│                                  │       O Create pull request options...                                                  #
│                                  │   <c-y> Copy pull request URL to clipboard                                              #
│                                  │       c Checkout by name                                                                #
│                                  │       F Force checkout                                                                  #
│                                  │       d Delete branch                                                                   #
│                                  │       r Rebase checked-out branch onto this branch                                      #
│                                  │       M Merge into currently checked out branch                                         #
└──────────────────────────────────│       f Fast-forward this branch from its upstream                                      #. (#1717)
┌Local branches - Remotes - Tags───│       T Create tag                                                                      #
│  * main ^2                       │       g View reset options...                                                           #
│3d  nvchad_restyle ?              │       R Rename branch                                                                   #.github.com>
│                                  │       u Set/Unset upstream...                                                           #
│                                  │       w View worktree options...                                                        #
│                                  │ <enter> View commits                                                                    #
│                                  │       / Filter the current view by text                                                 #
│                                  │                                                                                         #
│                                  │   <c-r> Switch to a recent repo                                                         #
│                                  │  <pgup> Scroll up main panel                                                            #
│                                  │<pgdown> Scroll down main panel                                                          #
│                                  │       @ Open command log menu...                                                        #
│                                  │       } Increase the size of the context shown around changes in the diff view          #
│                                  │       { Decrease the size of the context shown around changes in the diff view          #
│                                  │       : Execute custom command                                                          #
└──────────────────────────────────│   <c-p> View custom patch options...                                                   │
┌Commits - Reflog──────────────────│       m View merge/rebase options...                                                    │
│9ab9ef6b am Merge branch 'main' of│       R Refresh                                                                         │
│dfcedafa SC Update 16-about-sytemd.│      + Next screen mode (normal/half/fullscreen)                                       │
│bc1f7680 tl Remove excess Spaces. U│      _ Prev screen mode                                                                 │
│5880a6b2 RL New Crowdin updates (#1│      ? Open menu                                                                        │
│bf5ae469 tl Add tree and stat comma│  <c-s> View filter-by-path options...                                                   │
│56593b50 Jo Update rockydocs_webdev│      W Open diff menu...                                                                │
│9f6aad35 ss Add new features to the│  <c-w> Toggle whether or not whitespace changes are shown in the diff view              │                              v
│38768ac7 ss final fix, new characte│      z Undo                                                                             v─────────────────────────────┘
│b53919da AL feat: nvchad book (#170└────────────────────────────────────────────────────────────────────────────1 of 50┘┌──────────────────────────────┐
│5d6df56d ss revert non-functioning subscript `~` (#171│ │You can hide/focus this panel by pressing '@'                                             │
│267a9edc ss Minor edits, xfce 9-minimal (#1708)        │ │                                                                                          │
│6f3afa29 ss Add new features (#1707)                   │ │Random tip: You can page through the items of a panel using ',' and '.'                   │
│6de33bc4 am test page for the new style - NvChad Guide │ │                                                                                          │
│9dc47180 al add ShyRain as a contributor for content (v│ │                                                                                          │
└────────────────────────────────────────────1 of 300───┘ │                                                                                          │
┌Stash──────────────────────────────────────────────────┐ │                                                                                          │
│                                                       │ │                                                                                          │
└──────────────────────────────────────────────0 of 0───┘ └──────────────────────────────────────────────────────────────────────────────────────────┘
<enter>: Execute, <esc>: Close                                                                                        Donate Ask Question 0.40.2
```

Now that we have all the necessary supporting software on our system, we can move on to installing the basic software. We will start with the editor on which the whole configuration is based, Neovim.

4. Install Neovim

4.1 Introduction to Neovim

Neovim is one of the best code editors due to its speed, ease of customization, and configuration.

Neovim is a fork of the Vim editor. It was born in 2014, mainly due to the lack at the time of asynchronous job support in Vim. Written in the Lua language with the goal of modularizing the code to make it more manageable, Neovim was designed with the modern user in mind. As the official website states

Neovim is built for users who want the best parts of Vim, and more.

The developers of Neovim chose Lua as it was perfect for embedding, using LuaJIT quickly, and with a simple, script-oriented syntax.

From version 0.5 Neovim includes Treesitter (a parser generator tool) and supports Language Server Protocol (LSP). This reduces the number of plugins needed to achieve advanced editing functions. It improves the performance of operations such as code completion and linting.

One of its strengths is its customization. All its configurations are contained in a single file that can be distributed to various installations through version control systems (Git or other) so that they are always synchronized.

4.1.1 Community of developers

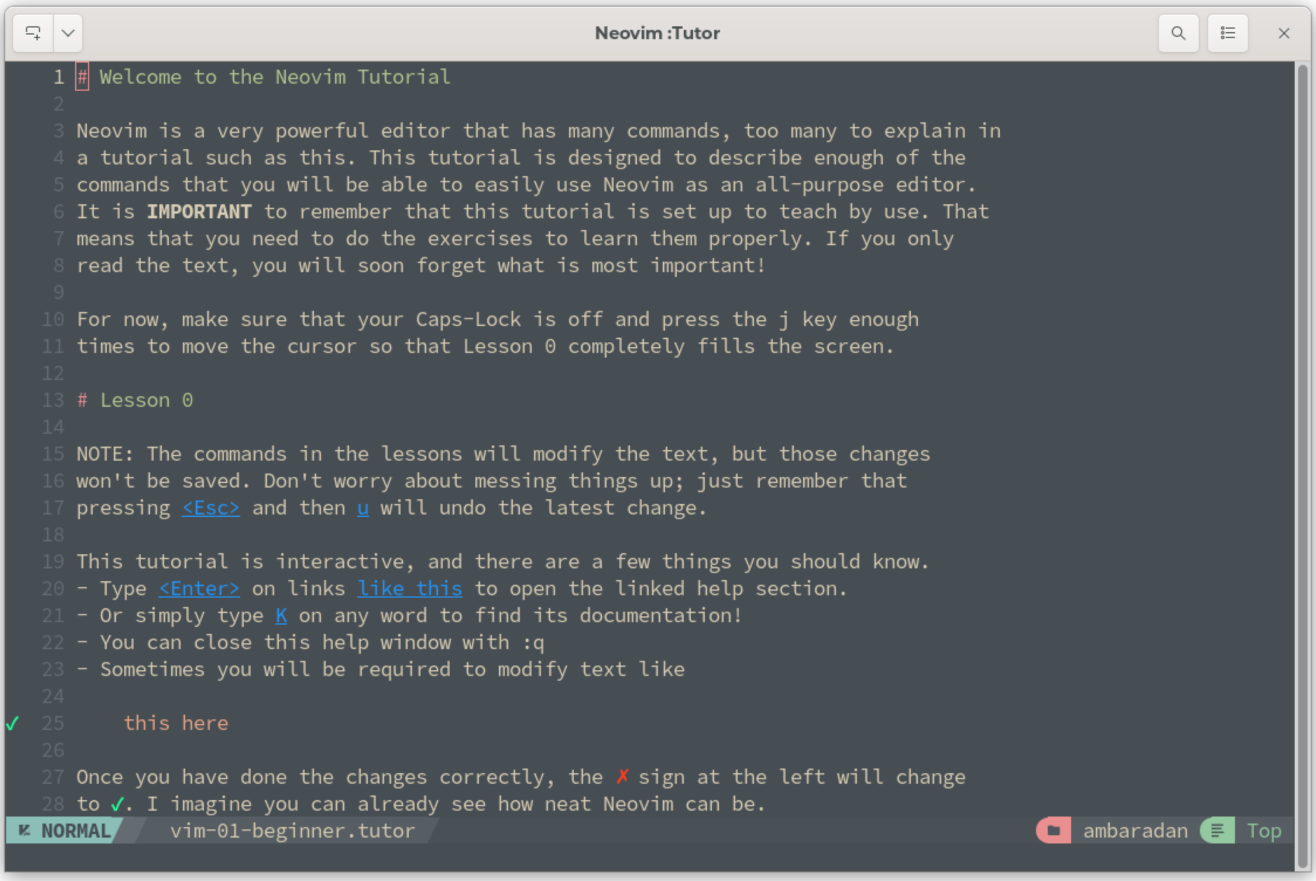
Although Vim and Neovim are both open-source projects and hosted on GitHub, there is a significant difference between the modes of development. Neovim has a more open community development, while Vim's development is more tied to the choices of its creator. Neovim's user and developer base is quite small compared to Vim, but it is a continuously growing project.

4.1.2 Key Features

- Performance: Very fast.

- Customizable: Wide ecosystem of plugins and themes
- Syntax highlighting: Integrated with Treesitter and LSP, but requires some configuration

As with Vim, Neovim requires a basic knowledge of its commands and options. You can get an overview of its features through the `:Tutor` command that invokes a file where you can read, and practice using it. Learning takes longer than a fully graphical IDE, but once you learn the shortcuts to the commands and the included features, you will proceed very smoothly in editing documents.



```
# Welcome to the Neovim Tutorial

Neovim is a very powerful editor that has many commands, too many to explain in
a tutorial such as this. This tutorial is designed to describe enough of the
commands that you will be able to easily use Neovim as an all-purpose editor.
It is IMPORTANT to remember that this tutorial is set up to teach by use. That
means that you need to do the exercises to learn them properly. If you only
read the text, you will soon forget what is most important!

For now, make sure that your Caps-Lock is off and press the j key enough
times to move the cursor so that Lesson 0 completely fills the screen.

# Lesson 0

NOTE: The commands in the lessons will modify the text, but those changes
won't be saved. Don't worry about messing things up; just remember that
pressing <Esc> and then u will undo the latest change.

This tutorial is interactive, and there are a few things you should know.
- Type <Enter> on links like this to open the linked help section.
- Or simply type K on any word to find its documentation!
- You can close this help window with :q
- Sometimes you will be required to modify text like

✓    this here

Once you have done the changes correctly, the ✗ sign at the left will change
to ✓. I imagine you can already see how neat Neovim can be.
 NORMAL   vim-01-beginner.tutor                          ambaradan   Top
```

4.2 Neovim Installation

Installation from EPEL

Neovim is also installable from the EPEL repository. The available version is always too old to meet the minimum requirements of the NvChad installation.
Installation by this method is strongly discouraged and is not supported in this guide.

Installation from pre-compiled package

Use of the pre-compiled package allows installation of both the development and stable versions, which meet the requirements, and can be used as the basis for configuring NvChad.

To use the full functionality of the editor, it is necessary to satisfy the dependencies required by Neovim by manually providing the pre-compiled package dependencies. The required packages can be installed with:

```
dnf install compat-lua-libs libtermkey libtree-sitter libvterm luajit
luajit2.1-luv msgpack unibilium xsel
```

After installing the required dependencies, it is time to acquire the chosen package.

By accessing the release page it will be possible to download the development version (pre-release) or the stable version (stable). In both cases the compressed archive to download for our architecture is linux64.

The required file is nvim-linux64.tar.gz, we should also download the file nvim-linux64.tar.gz.sha256sum to verify its integrity.

Assuming that both were downloaded to the same folder, we will use the following command for verification:

```
sha256sum -c nvim-linux64.tar.gz.sha256sum
nvim-linux64.tar.gz: OK
```

Now unpack the precompiled package to a location within your home folder, in this guide the location `.local/share/` was chosen but can be changed according to your needs. Run the command:

```
tar xvzf nvim-linux64.tar.gz -C ~/.local/share/
```

All that remains at this point is to create a symbolic link in ~/.local/bin/ for the nvim executable of the precompiled package.

```
cd ~/.local/bin/
ln -sf ~/.local/share/nvim-linux64/bin/nvim nvim
```

To verify the correct installation run in a terminal the command `nvim -v`, which should now show something like: