



NvChad (Ukrainian version)











A book from the Documentation Team

Version : 2024/04/29

Rocky Documentation Team

Copyright © 2023 The Rocky Enterprise Software Foundation

Table of contents

1.	Ліцензія	7
2.	 Вступ	8
2.1	 Використання Neovim як IDE	9
2.1.1	Основні особливості	10
2.2	Посилання	12
2.2.1	 Lua	12
2.2.2	 Neovim	14
2.2.3	 LSP	14
2.2.4	 tree-sitter	14
3.	 Необхідне додаткове програмне забезпечення	16
3.1	 RipGrep	16
3.2	 Перевірка RipGrep	17
3.3	 Lazygit	18

4.		Встановлення Neovim	21
4.1		Вступ до Neovim	21
4.1.1			
			
		Спільнота розробників	21
4.1.2		Ключові особливості	22
4.2		Інсталяція Neovim	23
4.2.1		Видалення	24
4.3		Neovim Basic	0
5.		Перетворення Neovim на просунутий IDE	0
5.1		Передумови	0
5.1.1		Попередні операції	0

5.2	 Встановлення	0
5.2.1	 Bootstrap	0
5.3	 Структура конфігурації	0
5.4	 Основні клавіші клавіатури	0
6.	Приклад конфігурації	0
6.1	 Вступ	0
6.2	 Встановлення	0
6.3	 Структура	0
6.4	 Аналіз структури	0
6.4.1	 Основні файли	0
6.4.2	 Папка configs	0
6.5	 Висновок	0
7.	 Nerd Fonts - Шрифти для розробників	0
7.1	 Завантаження	0
7.2	 Інсталяція	0
7.3	 Конфігурація	0
8.	vale в NvChad (Neovim)	0
8.1	 Вступ	0
8.2	 Передумови	0
8.2.1	 Встановлення nvim-lint	0
8.3	 Встановлення vale за допомогою Mason	0
8.3.1	 Налаштування та ініціалізація vale	0
8.3.2	 Зміна файлу lint.lua	0
8.4	Висновки та заключні думки	0
9.	Marksman - помічник коду	0
9.1	Цілі	0
9.2	Вимоги та навички	0
9.3	Установка Marksman	0

9.4	Інтеграція в редактор	0
9.5	Використання marksman	0
9.6	Основні клавіші	0
9.7	Висновок	
10.	Плагіни базової конфігурації	0
10.1	Основні плагіни	0
10.2	Функціональність LSP	0
10.3	Код Lua	0
10.4	Змішані плагіни	0
10.5	Керування файлами	0
11.	Менеджер плагінів	0
11.1	Основні особливості	0
11.2	Попередні операції	0
11.3	Вставити плагін	0
11.4	Видалення плагіна	0
11.5	Оновлення плагінів	0
11.6	Додаткові можливості	0
11.7	Синхронізація	0
12.	Інтерфейс NvChad	0
12.1	Tabufline	0
12.2	Середній розділ - відкриті буфери	0
12.3	Statusline	0
12.4	Інтегрована довідка	0
12.5	NvimTree	0
13.	Редагування за допомогою NvChad	0
13.1	Відкрити файл	0
13.2	Робота з редактором	0
13.2.1	Виділення тексту	0
13.2.2	Текстовий пошук	0
13.3	Збереження документа	0
14.	NvimTree - Провідник файлів	0
14.1	Робота з файловим провідником	0
14.1.1	Обрати файл	0
14.1.2	Відкрити файл	0
14.1.3	Керування файлами	0
14.2	Додаткові функції	0
14.3	Висновок	0

15. Огляд Markdown	0
15.1 Вступ	0
15.1.1 Peek.nvim	0
15.1.2 Markdown-preview.nvim	0
15.2 Висновки та заключні думки	0
16. Менеджер проекту	0
16.1 Вступ	0
16.1.1 Установка плагіна	0
16.1.2 Використання менеджера проекту	0
16.1.3 Додаткові функції	0
16.1.4 Маппінг	0
16.2 Висновки та заключні думки	0

1. Ліцензія

RockyLinux пропонує курси Linux для викладачів або людей, які бажають навчитися самостійно адмініструвати систему Linux.

Матеріали RockyLinux опубліковані під Creative Commons-BY-SA. Ви можете ділитися та трансформувати матеріал, дотримуючись прав автора.

BY : Attribution. Необхідно вказати ім'я оригінального автора.

SA : Share Alike.

- Ліцензія Creative Commons-BY-SA: <https://creativecommons.org/licenses/by-sa/4.0/>

Документи та їх джерела можна безкоштовно завантажити з:

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

Наші медіаджерела розміщені на github.com. Ви знайдете сховище вихідного коду, де було створено версію цього документа.

Ви можете створити персоналізований навчальний матеріал із цих джерел за допомогою [mkdocs](#). Інструкції щодо створення документа ви знайдете [тут](#).

Як я можу зробити внесок у проект документації?

Ви знайдете всю необхідну інформацію, щоб приєднатися до нас, на [домашній сторінці проекту git](#).

Бажаємо всім приємного читання та сподіваємося, що вам сподобається вміст.

2. Вступ

Зміни у випуску 2.5

З випуском версії 2.5 розробники редактора суттєво змінили структуру конфігурації. Найбільш істотні зміни стосуються таких аспектів:

- Перетворення конфігурації на плагін Neovim можна оновити за допомогою менеджера плагінів *lazy.nvim*
- Видалення папки `custom` для налаштування редактора (тепер інтегрована в основну папку). Для поточних користувачів надається [сценарій міграції](#).
- Зіставлення було змінено, і більше не використовується спеціальний синтаксис `nvchad`, замість нього використовується **`vim.keymap.set`** nvim.

У результаті деякі сторінки посібника, особливо вся частина, що стосується встановлення NvChad і подальшого встановлення плагінів, виглядають неправильно. Посібник **переглядається** та незабаром буде оновлено.

У цій книзі ви знайдете способи впровадження Neovim разом із NvChad для створення повнофункціонального **Integrated Development Environment** (IDE).

Я кажу «шляхи», тому що є багато можливостей. Тут автор зосереджується на використанні цих інструментів для написання markdown, але якщо markdown не є вашою метою, не хвилюйтеся, просто читайте далі. Якщо ви не знайомі з жодним із цих інструментів (NvChad або Neovim), тоді ця книга дасть вам вступ до обох, і якщо ви покроково ознайомитеся з цими документами, ви незабаром зрозумієте, що можете налаштувати це середовище, щоб воно було величезною допомогою для будь-яких потреб у програмуванні чи написанні сценаріїв.

Хочете IDE, яка допоможе писати playbooks Ansible? Ви можете це отримати! Хочете IDE для Golang? Це теж доступно. Вам потрібен хороший інтерфейс для написання сценаріїв BASH? Це також доступно.

Через використання **Language Server Protocols** і **linters**, ви можете налаштувати середовище, налаштоване саме для вас. Найкраще те, що після налаштування середовища його можна швидко оновити, коли з'являться нові зміни, за допомогою *lazy.nvim* і *Mason*, обидва з яких є розглянуто тут.

Оскільки Neovim є розгалуженням *Vim*, загальний інтерфейс буде знайомий для *vim* користувачів. Якщо ви не є користувачем *vim*, ви швидко зрозумієте синтаксис команд за допомогою цієї книги. Тут міститься багато інформації, але легко сприймається, і коли ви завершите вивчення вмісту, ви будете знати

достатньо, щоб створити власну IDE для *своїх* потреб за допомогою цих інструментів.

Автор **не** мав наміру розбивати цю книгу на розділи. Причина в тому, що це передбачає порядок, якого потрібно дотримуватися, і, здебільшого, це не обов'язково. Ви *можете* почати з цієї сторінки, прочитати та дотримуватися розділів «Додаткове програмне забезпечення», «Установити Neovim» і «Установити NvChad», але звідти ви можете вибрати як ви хочете продовжити.

2.1 Використання Neovim як IDE

Базова інсталяція Neovim надає чудовий редактор для розробки, але його ще не можна назвати IDE; усі розширені функції IDE, навіть якщо вони вже встановлені, ще не активовані. Для цього нам потрібно передати необхідні конфігурації Neovim, і тут нам на допомогу приходить NvChad. Це дозволяє нам мати базову конфігурацію з коробки лише однією командою!

Конфігурація написана на Lua, дуже швидкій мові програмування, яка дозволяє NvChad мати дуже швидкий час запуску та виконання для команд і натискань клавіш. Це також стало можливим завдяки техніці **Lazy loading**, яка використовується для плагінів і завантажує їх лише за потреби.

Інтерфейс виходить дуже чистим і приємним.

Як зазначають розробники NvChad, проект призначений лише як основа для створення вашої власної персональної IDE. Подальше налаштування здійснюється за допомогою плагінів.

```

1  |
2  title: Home
3  ---
4
5  # Rocky Linux Documentation
6
7  ## Welcome!
8
9  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?
10
11 Well actually, you do.
12
13 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.
14
15 ## Navigating the Site
16
17 ### Getting Around
18
19 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.

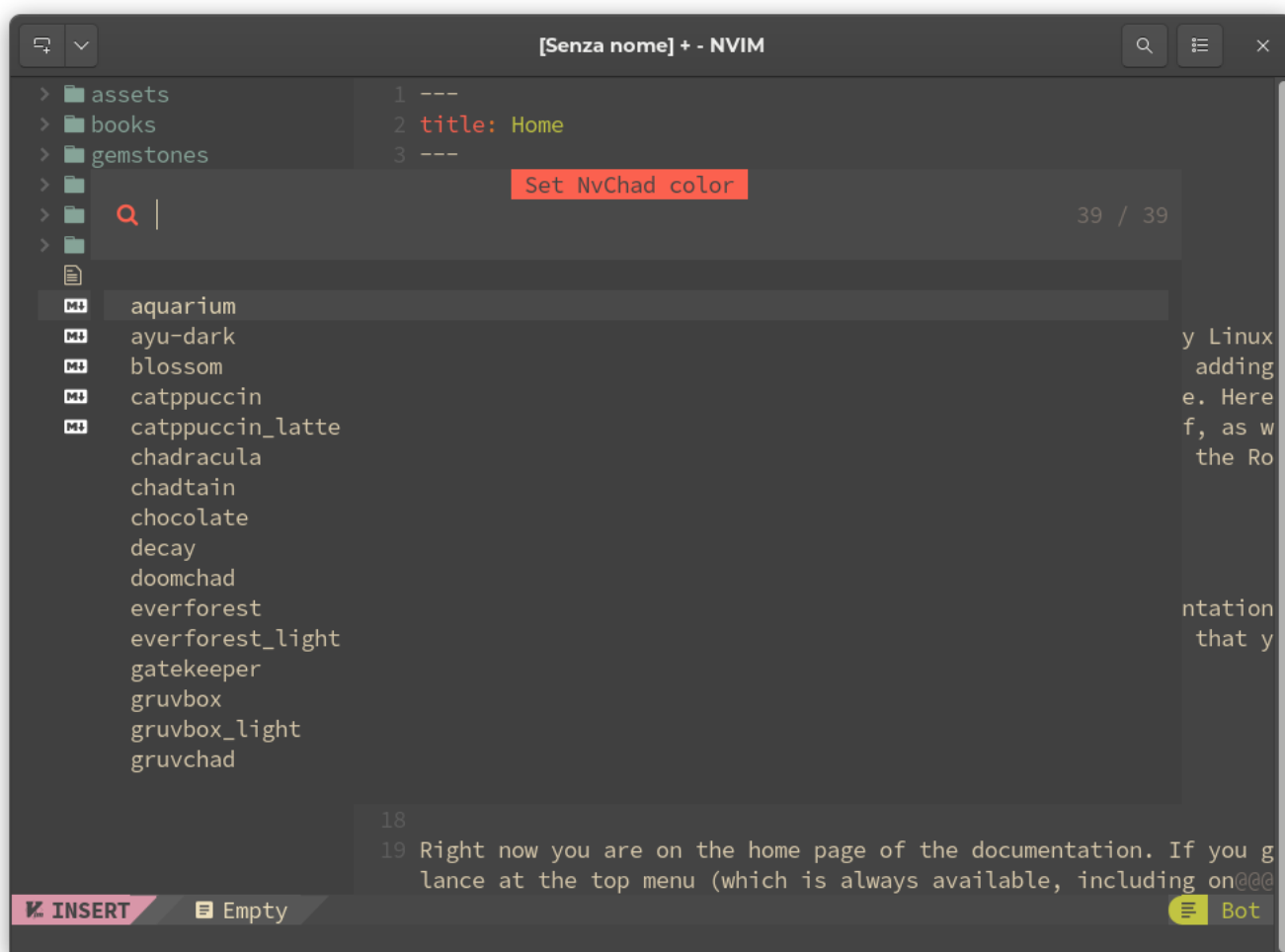
```

NORMAL index.en.md main LSP documentation Top

2.1.1 Основні особливості

- **Розроблено, щоб бути швидким.** Від вибору мови програмування до методів завантаження компонентів, усе створено для мінімізації часу виконання.
- **Привабливий інтерфейс.** Незважаючи на те, що це програма *cli*, інтерфейс виглядає сучасно та красиво графічно, а всі компоненти підходять інтерфейс ідеально.
- **Надзвичайна конфігурація.** Завдяки модульності, отриманій від базової програми (NeoVim), редактор можна ідеально адаптувати до потреб. Однак пам'ятайте, що коли ми говоримо про налаштування, ми маємо на увазі функціональність, а не зовнішній вигляд інтерфейсу.

- **Механізм автоматичного оновлення.** Редактор постачається з механізмом (через використання *git*), який дозволяє оновлення за допомогою проста команда `:NvChadUpdate`.
- **На базі Lua.** Конфігурація NvChad повністю написана мовою *lua*, що дає змогу легко інтегрувати її в конфігурацію Neovim за допомогою всього потенціалу редактора, на якому він заснований.
- **Безліч вбудованих тем.** Конфігурація вже містить велику кількість тем для використання, завжди пам'ятаючи, що ми говоримо про * програма cli*; теми можна вибрати за допомогою `<leader> +` клавіша `th`.



2.2 Посилання

2.2.1 Lua

Що таке Lua?

Lua — це надійна, легка мова сценаріїв, яка підтримує різноманітні методи програмування. Назва "Lua" походить від португальського слова, що означає «місяць.»

Lua було розроблено в Католицькому університеті Ріо-де-Жанейро Роберто Єрусалімським, Луїсом Енріке де Фігейредо та Вальдемаром Селесом. Розробка була необхідна для них, оскільки до 1992 року в Бразилії діяли суворі правила щодо імпорту апаратного та програмного забезпечення, тож із чистої необхідності ці троє програмістів розробили власну мову сценаріїв під назвою Lua.

Оскільки Lua зосереджена насамперед на сценаріях, вона рідко використовується як окрема мова програмування. Замість цього вона найчастіше використовується як мова сценаріїв, яку можна інтегрувати (вбудовувати) в інші програми.

Lua використовується при розробці відеоігор та ігрових движків (Roblox, Warframe..), як мова програмування в багатьох мережевих програмах (Nmap, ModSecurity..) і як мова програмування в промислових програмах. Lua також використовується як бібліотека, яку розробники можуть інтегрувати у свої програми, щоб увімкнути функціональність сценаріїв, діючи виключно як невід’ємна частина основної програми.

Як працює Lua

Є два основних компоненти Lua:

- Інтерпретатор Lua
- Віртуальна машина Lua (VM)

Lua не інтерпретується безпосередньо через файл Lua, як інші мови, наприклад Python. Натомість вона використовує інтерпретатор Lua для

компіляції файлу Lua у байт-код. Інтерпретатор Lua дуже портативний і здатний працювати на багатьох пристроях.

Ключові особливості

- **Швидкість:** Lua вважається однією з найшвидших мов програмування серед інтерпретованих мов сценаріїв; він може виконувати дуже важкі завдання швидше, ніж більшість інших мов програмування.
- **Розмір:** Lua крихітна порівняно з іншими мовами програмування. Цей невеликий розмір ідеально підходить для інтеграції Lua в різні платформи, від вбудованих пристроїв до ігрових движків.
- **Портативність та інтеграція:** портативність Lua практично необмежена. Будь-яка платформа, яка підтримує стандартний компілятор C, може без проблем запускати Lua. Lua не вимагає складних перезаписів, щоб бути сумісним з іншими мовами програмування.
- **Простота:** Lua має простий дизайн, але забезпечує потужну функціональність. Однією з головних особливостей Lua є метамеханізми, які дозволяють розробникам реалізувати їх функціональні можливості. Синтаксис простий і зрозумілий, тому будь-хто може легко вивчити Lua та використовувати його у своїх програмах.
- **Ліцензія:** Lua — це безкоштовне програмне забезпечення з відкритим вихідним кодом, що розповсюджується за ліцензією MIT. Це дозволяє будь-кому використовувати його без сплати ліцензії чи роялті.

2.2.2 Neovim

Neovim детально описано на [спеціальній сторінці](#), тому ми зупинимось лише на основних функціях, а саме:

- Продуктивність: дуже швидка.
- Можливість налаштування: широка екосистема плагінів і тем.
- Підсвічування синтаксису: інтеграція з Treesitter і LSP (вимагає деяких додаткових конфігурацій).
- Мультиплатформенність: Linux, Windows і macOS
- Ліцензія: MIT: коротка та проста дозвільна ліцензія з умовами, що вимагають лише збереження авторських прав і повідомлень про ліцензію.

2.2.3 LSP

Що таке **L**anguage **S**erver **P**rotocol?

Мовний сервер — це стандартизована мовна бібліотека, яка використовує власну процедуру (протокол) для підтримки таких функцій, як автозаповнення, визначення переходу або визначення наведення курсора.

Ідея протоколу мовного сервера (LSP) полягає в стандартизації протоколу зв'язку між інструментами та серверами, щоб один мовний сервер можна було повторно використовувати в кількох інструментах розробки. Таким чином, розробники можуть просто інтегрувати ці бібліотеки у свої редактори та посилатися на існуючі мовні інфраструктури, замість того, щоб налаштовувати свій код для їх включення.

2.2.4 tree-sitter

Tree-sitter в основному складається з двох компонентів: **генератора аналізатора** та **бібліотеки інкрементального аналізу**. Він може побудувати синтаксичне дерево вихідного файлу та ефективно оновлювати його з кожною зміною.

Синтаксичний аналізатор — це компонент, який розкладає дані на менші елементи, щоб полегшити їх переклад іншою мовою або, як у нашому випадку,

для передачі в бібліотеку аналізу. Після розкладання вихідного файлу бібліотека синтаксичного аналізу аналізує код і перетворює його на синтаксичне дерево, що дозволяє більш розумно керувати структурою коду. Це дає змогу покращити (і прискорити)

- підсвічування синтаксису
- навігацію по коду
- рефакторинг
- текстові об'єкти та рухи

Взаємодоповнюваність LSP і tree-sitter.

Хоча може здатися, що дві служби (LSP і tree-sitter) є зайвими, вони насправді доповнюють один одного, оскільки LSP працює на рівні проекту, тоді як tree-sitter працює лише з файлом з відкритим кодом.

Тепер, коли ми трохи пояснили технології, які використовуються для створення IDE, ми можемо перейти до [Додаткового програмного забезпечення](#), необхідного для налаштування нашого NvChad.

3. Необхідне додаткове програмне забезпечення

Є додаткове програмне забезпечення, яке, хоча і не потрібне, допоможе у загальному використанні NvChad. У наведених нижче розділах описано це програмне забезпечення та його використання.

3.1 RipGrep

`ripgrep` — це рядково-орієнтований інструмент пошуку, який рекурсивно шукає в поточному каталозі шаблон *regex* (регулярний вираз). За замовчуванням *ripgrep* дотримується правил *gitignore* і автоматично пропускає приховані файли/каталоги та двійкові файли. Ripgrep пропонує відмінну

підтримку для Windows, macOS і Linux, з двійковими файлами, доступними для кожного випуску.

Встановлення RipGrep з EPEL

У Rocky Linux 8 і 9 ви можете встановити RipGrep з EPEL. Для цього встановіть `epel-release`, оновіть систему, а потім встановіть `ripgrep`:

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

Встановлення RipGrep за допомогою cargo

Ripgrep — це програмне забезпечення, написане мовою *Rust*, яке можна встановити за допомогою утиліти `cargo`. Зауважте, однак, що `cargo` не встановлюється за замовчуванням *rust*, тому ви повинні встановити його явно. Якщо під час використання цього методу виникають помилки, поверніться до встановлення з EPEL.

```
dnf install rust cargo
```

Після встановлення необхідного програмного забезпечення ми можемо встановити `ripgrep` за допомогою:

```
cargo install ripgrep
```

Встановлення збереже виконуваний файл `rg` у папці `~/.cargo/bin`, яка знаходиться поза ШЛЯХОМ, щоб використовувати його на рівні користувача, ми зв'яжемо його з `~/.local/bin/`.

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

3.2 Перевірка RipGrep

На цьому етапі ми можемо перевірити, чи все в порядку з:

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

RipGrep потрібен для рекурсивного пошуку за допомогою `:Telescope`.

3.3 Lazygit

LazyGit — це інтерфейс у стилі ncurses, який дозволяє виконувати всі операції `git` у більш зручній для користувача формі спосіб. Це потрібно для плагіна `lazygit.nvim`. Цей плагін дає змогу використовувати LazyGit безпосередньо з NvChad, він відкриває плаваюче вікно, з якого ви можете виконувати всі операції зі своїми репозиторіями, таким чином дозволяючи вам вносити всі зміни до *репозиторію git*, не виходячи з редактора.

Щоб встановити його, ми можемо скористатися репозиторієм для Fedora. На Rocky Linux 9 це працює ідеально.

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

Після встановлення ми відкриваємо термінал і вводимо команду `lazygit`, і з'являється інтерфейс, схожий на цей:

The screenshot shows the Lazygit interface with the following panels:

- Status:** Shows the current branch as `documentation` and the commit hash `9ab9ef6b`.
- Files:** Shows the file `nvchad_restyle ?` with a status of `3d`.
- Local branches:** Shows the `main` branch with a status of `2`.
- Commits:** Shows a list of commits, including the current one `9ab9ef6b` and previous ones like `dfcedafa`, `bc1f7680`, and `5880a6b2`.
- Log:** Shows the commit history, including the merge of `main` into `documentation` and the update of `nvchad_restyle`.
- Command log:** Shows the command `pgup` being used to scroll up.

The interface also includes a status bar at the bottom with keyboard shortcuts and a footer with the text `Copyright © 2023 The Rocky Enterprise Software Foundation`.

За допомогою клавіші ми можемо викликати меню з усіма доступними командами.

```

Status
~2 documentation > main

Files - Worktrees - Submodules

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

Commits - Reflog
9ab9ef6b am Merge branch 'main' of
dfcedafa SC Update 16-about-sytemd.
bc1f7680 tl Remove excess Spaces. U
5880a6b2 RL New Crowdin updates (#1
bf5ae469 tl Add tree and stat comma
56593b50 Jo Update rockydocs webdev
9f6aad35 ss Add new features to the
38768ac7 ss final fix, new characte
b53919da AL feat: nvchad book (#170
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

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
T Create tag
g View reset options...
R Rename branch
u Set/Unset upstream...
w View worktree options...
<enter> View commits
A 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...
m View merge/rebase options...
R Refresh
+ Next screen mode (normal/half/fullscreen)
_ Prev screen mode
? Open menu
<c-s> View filter-by-path options...
W Open diff menu...
<c-w> Toggle whether or not whitespace changes are shown in the diff view
z Undo

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

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

1 of 50

<enter>: Execute, <esc>: Close
Donate Ask Question 0.40.2

```

Тепер, коли в нашій системі є все необхідне допоміжне програмне забезпечення, ми можемо переходити до встановлення основного програмного забезпечення. Ми почнемо з редактора, на якому базується вся конфігурація, [Neovim](#).

4. Встановлення Neovim

4.1 Вступ до Neovim

Neovim є одним із найкращих редакторів коду завдяки своїй швидкості, простоті налаштування та конфігурації.

Neovim є відгалуженням редактора **Vim**. Він народився в 2014 році, в основному через відсутність на той час підтримки асинхронних завдань у Vim. Написаний мовою **Lua** з метою модуляризації коду, щоб зробити його більш керованим, Neovim був розроблений з урахуванням сучасного користувача. Як повідомляє офіційний сайт

Neovim створено для користувачів, яким потрібні найкращі частини Vim тощо.

Розробники Neovim обрали Lua, оскільки вона ідеально підходить для вбудовування, швидко використовує LuaJIT і має простий синтаксис, орієнтований на сценарії.

Починаючи з версії 0.5, Neovim включає **Treesitter** (інструмент генератора парсерів) і підтримує **Language Server Protocol** (LSP). Це зменшує кількість плагінів, необхідних для досягнення розширених функцій редагування. Це покращує продуктивність таких операцій, як завершення коду та лінтування.

Однією з його сильних сторін є можливість налаштування. Усі його конфігурації містяться в одному файлі, який можна розповсюджувати на різні інсталяції через системи контролю версій (Git або інші), щоб вони завжди були синхронізовані.

4.1.1 Спільнота розробників

Хоча Vim і Neovim є проектами з відкритим вихідним кодом і розміщені на GitHub, існує значна різниця між способами розробки. Neovim має більш відкритий розвиток спільноти, тоді як розвиток Vim більше прив'язаний до вибору його творця. База користувачів і розробників Neovim досить мала порівняно з Vim, але це проект, який постійно зростає.

4.1.2 Ключові особливості

- Продуктивність: дуже швидка.
- Можливість налаштування: широка екосистема плагінів і тем
- Підсвічування синтаксису: інтегровано з Treesitter і LSP, але вимагає певної конфігурації

Як і у випадку з Vim, Neovim вимагає базових знань про його команди та параметри. Ви можете отримати огляд його можливостей за допомогою команди `:Tutor`, яка викликає файл, який ви можете прочитати та потренуватися використовувати його. Навчання займає більше часу, ніж повна графічна IDE, але як тільки ви вивчите ярлики команд і включені функції, ви зможете дуже легко редагувати документи.

```

1 # Welcome to the Neovim Tutorial
2
3 Neovim is a very powerful editor that has many commands, too many to explain in
4 a tutorial such as this. This tutorial is designed to describe enough of the
5 commands that you will be able to easily use Neovim as an all-purpose editor.
6 It is IMPORTANT to remember that this tutorial is set up to teach by use. That
7 means that you need to do the exercises to learn them properly. If you only
8 read the text, you will soon forget what is most important!
9
10 For now, make sure that your Caps-Lock is off and press the j key enough
11 times to move the cursor so that Lesson 0 completely fills the screen.
12
13 # Lesson 0
14
15 NOTE: The commands in the lessons will modify the text, but those changes
16 won't be saved. Don't worry about messing things up; just remember that
17 pressing <Esc> and then u will undo the latest change.
18
19 This tutorial is interactive, and there are a few things you should know.
20 - Type <Enter> on links like this to open the linked help section.
21 - Or simply type K on any word to find its documentation!
22 - You can close this help window with :q
23 - Sometimes you will be required to modify text like
24
25 ✓      this here
26
27 Once you have done the changes correctly, the X sign at the left will change
28 to ✓. I imagine you can already see how neat Neovim can be.

```

ambaradan Top

4.2 Інсталяція Neovim

Інсталяція з EPEL

Neovim також можна встановити з репозиторію EPEL. Доступна версія завжди занадто стара, щоб відповідати мінімальним вимогам інсталяції NvChad.

Встановлення цим методом настійно не рекомендується та не підтримується в цьому посібнику.

Встановлення з попередньо скомпільованого пакета

Використання попереднього пакета дозволяє встановити як розробну, так і стабільну версії, які відповідають вимогам і можуть бути використані як основа для налаштування NvChad.

Щоб використовувати повну функціональність редактора, необхідно задовольнити залежності, які вимагає Neovim, вручну надавши попередньо скомпільовані залежності пакета. Необхідні пакети можна встановити за допомогою:

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

Після встановлення необхідних залежностей настав час придбати вибраний пакет.

Перейшовши на [сторінку випуску](#), ви зможете завантажити версію для розробки ([pre-release](#)) або стабільну версію ([stable](#)). В обох випадках стислий архів для завантаження для нашої архітектури [linux64](#).

Необхідний файл [nvim-linux64.tar.gz](#); ми також повинні завантажити файл [nvim-linux64.tar.gz.sha256sum](#), щоб перевірити його цілісність.

Припускаючи, що обидва були завантажені в одну папку, ми використаємо таку команду для перевірки:

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

Тепер розпакуйте попередньо скомпільований пакет до місця у вашій домашній папці; у цьому посібнику вибрано розташування `~/.local/share/`, але його можна змінити відповідно до ваших потреб. Виконайте команду:

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

Все, що залишається на цьому етапі, це створити символічне посилання в `~/.local/bin/` для виконуваного файлу `nvim` попередньо скомпільованого пакета.

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

Щоб перевірити правильність встановлення, запустіть у терміналі команду

`nvim` у яка тепер має показати щось на зразок: