CONFIGURING NEOVIM AS A *PERSONAL DEVELOPMENT ENVIRONMENT (PDE)* TO FULL STACK DEVELOPMENT

Paulo Jerônimo

COPY AND PASTE BASH/LUA CODES TO YOUR TERMINAL TO LEARN HOW TO CREATE YOUR PDE QUICKLY!

Date time / Last commit: 2023-06-10 13:08:33 -0300 / 7804317
Online version: https://paulojeronimo.com/tutorials/neovim.pdf
About me (the author)

In this tutorial (***UNDER DEVELOPMENT***) I'll explain how to configure Neovim with Lua to do **Full Stack Development** with Java, Kotlin, and some other languages.

Watch me execute this entire tutorial on this video.

At the end of this tutorial, you will have a Git repository (with four (4) commits) with all the necessary settings to have a **Personal Development Environment (PDE)**. A PDE is an **Integrated Development Environment (IDE)**, completely customized to suit your way of developing!

Some tips ...

1. As in my previous tutorial (links below) **we'll follow some steps by copying and pasting Bash command lines to an opened terminal. Super easy ☺!!!**

   Previous tutorial:
   https://paulojeronimo.com/asciidoctor-web3-article/private/sample.pdf
   YouTube video following it (spoken in Portuguese):
   https://www.youtube.com/watch?v=UCac0PziHHQ

2. You can **buy this tutorial (know the benefits)!** So you will be able to replicate this entire tutorial in your environment much more quickly than you will see me do in the Video version where I follow it!

Video version

TODO

Final Git repository

With the following Git command, we can see all the four (4) commits we will make throughout this tutorial. We will start from scratch after installing Neovim.

```
git log --pretty=format:"--> commit %h: %s <--" --name-status
```

Output:

```
--> commit bf225f9: Added treesitter and telescope plugins <--
M       lua/core/plugin_config/init.lua
A       lua/core/plugin_config/telescope.lua
A       lua/core/plugin_config/treesitter.lua
M       lua/core/plugins.lua

--> commit d387ac5: Installed and configured four (4) plugins <--
M       init.lua
A       lua/core/plugin_config/gruvbox.lua
A       lua/core/plugin_config/init.lua
A       lua/core/plugin_config/lualine.lua
A       lua/core/plugin_config/nvim-tree.lua
M       lua/core/plugins.lua

--> commit 175f515: Added packer.nvim <--
A       .gitignore
M       init.lua
A       lua/core/plugins.lua

--> commit 18da2bc: Created some very basic configurations <--
A       init.lua
A       lua/core/keymaps.lua
```

Buy this tutorial!

Why do that since I'm reading it for free?

For **six (6) months** from the time of your purchase of this tutorial, you will:

1. **Receive a non-rasterized version of this PDF file configured with your name (<- UNDER DEVELOPMENT)**. A rasterized PDF causes links to stop working and prevents text from being selected. So, after buying this tutorial, you will be able to:
   a. **Navigate to the internal (or external) links configured inside it**. Including the links to references shown in the Video version of this tutorial.
   b. **Copy and paste commands through links associated with code images** inside it, as you saw in this Video version.

> So, when buying this tutorial, instead of typing all the commands presented here manually, one by one, and with the possibility of making mistakes during this task, the payable version allows you to access the code by clicking on any image associated with one. After that, you can simply paste it on your terminal ☺!

2. **Receive any updates made on it**. Including any new PDF or video versions available of it.
3. **Access to the Portuguese-spoken videos executing this tutorial**. So, if you are interested in learning Portuguese (Brazilian accent) or if you want to listen to the videos of this tutorial in that language (if your native language is Portuguese, for example), you will have access to it.

4. **Receive direct support**. This means that I (Paulo Jerônimo) will solve your doubts as long as they are associated with the content presented in this tutorial.
5. **Receive direct access to the Final Git repository** to clone it.
6. **Receive Web3 Tokens** (<- UNDER DEVELOPMENT).

How much do I have to pay?

Only US $X.89 (X US dollars and eighty-nine cents) ☺!!!

Click here to buy! (<- UNDER DEVELOPMENT)

About me (the author)

My name is **Paulo Jerônimo**. I work as a Full Stack Developer who has been acting in several areas of software development (since 1993). I have experience from the infrastructure to the front end, passing through extensive coding time in middleware and the back end.

My main focus (on personal projects) has been developing applications that use technologies such as Web3 and Artificial Intelligence.

For example, related to Web3, I am currently developing solutions based on NFTs, including tokenizing my technical tutorials. I have also been developing solutions to tokenize the physical medals of participants in sports competitions.

Related to Artificial Intelligence, I am currently developing plugins for tools like the ChatGPT to create training plans for physical activities.

I'm a triathlete, three (3) times Ironman. I like to encourage people through my example in sports. Unfortunately, just demonstrating this example is not enough. Each person needs to find something, internally within himself, that motivates him to have daily physical activities.

When I do aerobic activities sometime after waking up (such as walking, running, swimming, or cycling), I increase my reasoning, learning, and concentration potential for the activities I will do during the day. Also, I'm easy to put on weight. So I worry about that and try not to fall into that problem. However, sometimes I get careless.

I'm forced to escape this damage when I don't care for my health. And I'm a guy who prefers to do things with pleasure rather than obligatory. So, once again, neglecting to care for my health is not an option.

By understanding all this and experiencing a disciplined routine in sports with the ultimate aim of healthy aging, I develop personal projects that help people acquire my way of acting.

In my projects, which I'm continuously developing using part of my free time, I try to connect my knowledge of technologies already consolidated in software development with the application of technologies that are now becoming even more popular: Web3 and Artificial Intelligence (like I said). My purpose in using these technologies is to solve a single problem: how do we encourage people to become more physically active to live happier and healthy?

I'm Brazilian. So English is not my primary language. But I'm sharing the most I can in English to immerse myself in writing and speaking in this language already some years.

Nowadays, I only write my tutorials, like this one, in English. And I have many of them written. You can see some in the following link:

https://paulojeronimo.com/sitemap/#articles-or-tutorials

My intention with this approach is also to help other software developers who speak Portuguese, like me, to improve their skills. I also write closed captions in Portuguese for my videos when I create them in English.

I'm also starting to produce videos in English, with subtitles in this language and Portuguese, at the same time.

Based on the list of technologies I mentioned (on this page) that I have some experience with, you will notice that I am a very versatile professional with skills in several languages, frameworks, tools, etc.

As I said, I have a focus (for my projects) these days. But, my experience as a developer in languages like Bash and Java is very long. Also, I have been expanding my skills in JavaScript, Python, and Kotlin for a few years.

In short, I don't consider myself an expert in anything today. But, I have a lot of aptitude for learning quickly and teaching what I know in a systematic and organized way (as you can see in the tutorials I write).

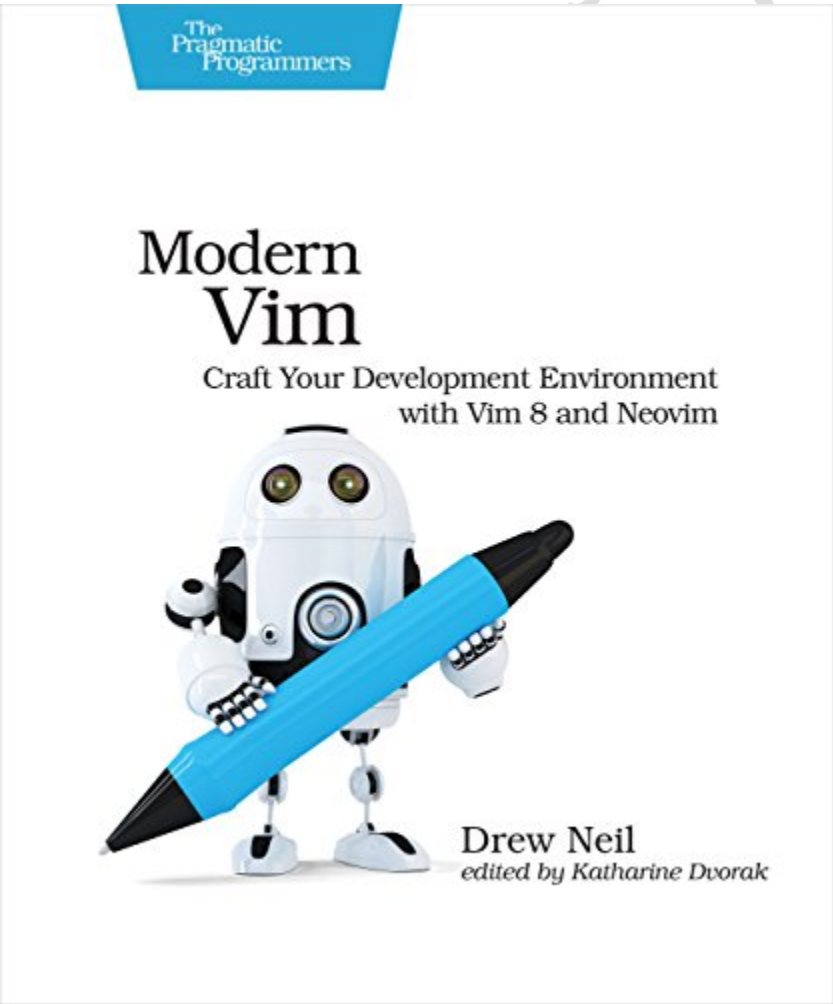
Today, my vision for Full Stack Developer professionals like me is that they will be able to benefit more and more in developing solutions from Artificial Intelligence tools. But, I will detail more in other texts than this one.

Table of Contents

1. Prerequisites to start

1.1. Basic Vim knowlegde

If you are new to Neovim (or Vim), or even if you have some experience on it, my recommendation is that you click the following image and purchase this book:

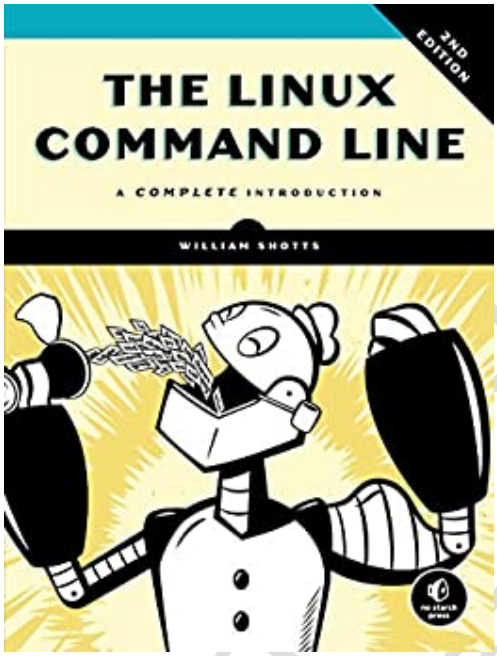


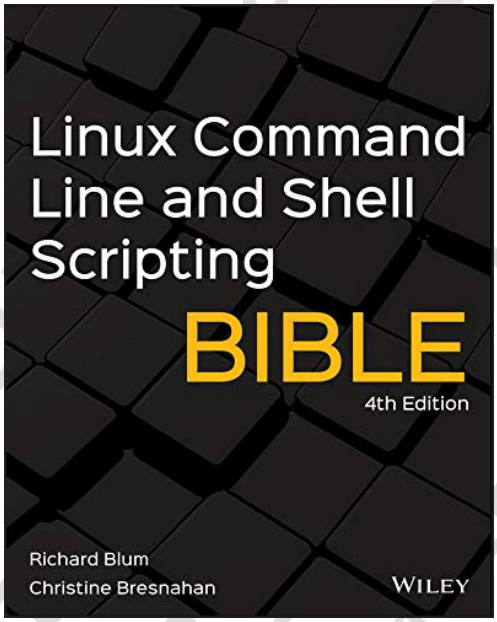
1.2. Basic Linux Command Line experience

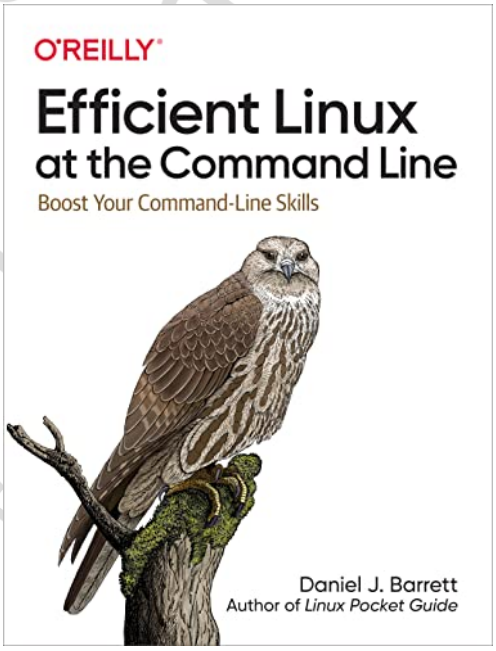
All the commands we'll follow in this tutorial will be copied to a Linux terminal and execute some actions, like creating or modifying files, directories, etc.

We'll use Bash to automate many things for us, and I assume you have some essential experience with the Linux Command Line.

Maybe you want to dive deep into this subject. In this case, I recommend the following books:







1.3. Ubuntu 22.04 installed

We'll be configuring Neovim, in this tutorial, in a Ubuntu 22.04 development environment.

Future versions of this tutorial will adapted to do the following configurations on macOS or Windows environments too. *Buying this tutorial let's you receive their updates*!

2. Let’s start …

As of this page, all links are disabled.
Buy this tutorial to get the most value out of it!

3. Creating a `nvim-dev` user

Let's create a new user and make the settings for it so as not to change your current environment in Ubuntu if you already have one set up.

Create a `nvim-dev` user:

```
sudo useradd -m -s /bin/bash -G sudo nvim-dev
```

Log in with it:

```
sudo su - nvim-dev
```

Configure git:

```
{
  git config --global user.name "Paulo Jerônimo"
  git config --global user.email paulojeronimo@gmail.com
  git config --global init.defaultBranch main
  git config --global diff.tool nvimdiff
  git config --global difftool.prompt false
}
```

Check your Git configurations and, if neeed redo any Git command line shown above:

```
git config --global -l
```

4. Installing Neovim

Install Neovim:

```
sudo apt install -y neovim
```

Check the version of it (should be `v0.8.3`):

```
nvim --version
```

Create the `vim` alias to the `nvim` command:

```
{
  echo 'alias vim=nvim' > ~/.bash_aliases
  source ~/.bash_aliases
}
```

That way, when calling `vim`, you are invoking `nvim`, which will help you migrate smoothly to it.

5. Creating a simple `nvim` configuration

Create a directory to create some Lua code:

```
mkdir -p ~/.config/nvim/lua/core && cd $_
```

Create the file `keymaps.lua`:

```
cat > keymaps.lua <<'EOF'
vim.opt.tabstop = 2
vim.opt.shiftwidth = 2
vim.opt.expandtab = true
EOF
```

Create the file `init.lua`:

```
cd ../.. && echo 'require("core.keymaps")' > init.lua
```

Do your first commit:

```
git init &&
git add -A &&
git commit -m 'Created some very basic configurations'
```

6. Installing `packer.nvim`

Create the file `lua/core/plugins.lua`:

```
cat > lua/core/plugins.lua <<'EOF'
local ensure_packer = function()
  local fn = vim.fn
  local install_path = fn.stdpath('data')..'/site/pack/packer/start/packer.nvim'
  if fn.empty(fn.glob(install_path)) > 0 then
    fn.system({'git', 'clone', '--depth', '1',
      'https://github.com/wbthomason/packer.nvim', install_path})
    vim.cmd [[packadd packer.nvim]]
    return true
  end
  return false
end

local packer_bootstrap = ensure_packer()

return require('packer').startup(function(use)
  use 'wbthomason/packer.nvim'
  -- My plugins here
  -- use 'foo1/bar1.nvim'
  -- use 'foo2/bar2.nvim'

  -- Automatically set up your configuration after cloning packer.nvim
  -- Put this at the end after all plugins
  if packer_bootstrap then
    require('packer').sync()
  end
end)
EOF
```

Add core.plugins to init.lua:

```
echo 'require("core.plugins")' >> init.lua
```

Configure an alias to `PackerSync` and invoke it:

```
{
  cat >> ~/.bash_aliases <<'EOF'
alias PackerSync='nvim --headless -c \
"autocmd User PackerComplete quitall" -c PackerSync'
EOF
  source ~/.bash_aliases
  PackerSync
}
```

Create .gitignore:

```
echo plugin > .gitignore
```

Do a new Git commit:

```
git add -A &&
git commit -m 'Added packer.nvim'
```

7. Installing and configuring four (4) simple plugins

Apply the following diff:

```
git apply <<'EOF'
diff --git a/lua/core/plugins.lua b/lua/core/plugins.lua
index 3a1ad95..bfe2bb3 100644
--- a/lua/core/plugins.lua
+++ b/lua/core/plugins.lua
@@ -13,9 +13,10 @@ local packer_bootstrap = ensure_packer()

 return require('packer').startup(function(use)
   use 'wbthomason/packer.nvim'
-  -- My plugins here
-  -- use 'foo1/bar1.nvim'
-  -- use 'foo2/bar2.nvim'
+  use 'ellisonleao/gruvbox.nvim' -- colorscheme
+  use 'nvim-tree/nvim-tree.lua' -- file explorer tree
+  use 'nvim-tree/nvim-web-devicons' -- icons
+  use 'nvim-lualine/lualine.nvim' -- status line

   -- Automatically set up your configuration after cloning packer.nvim
   -- Put this at the end after all plugins
EOF
```

Install the configured plugins:

```
PackerSync
```

Configure the installed plugins:

```
mkdir lua/core/plugin_config && cd $_
```

Create the file `grubbox.lua`:

```
cat > gruvbox.lua <<'EOF'
vim.o.termguicolors = true
vim.cmd [[ colorscheme gruvbox ]]
EOF
```

Create the file `lualine.lua`:

```
cat > lualine.lua <<'EOF'
require('lualine').setup {
  options = {
    icons_enabled = true,
    theme = 'gruvbox',
  },
  sections = {
    lualine_a = {
      {
        'filename',
        path = 1
      }
    }
  }
}
EOF
```

Create the file `nvim-tree.lua`:

```
cat > nvim-tree.lua <<'EOF'
vim.g.loaded_netrw = 1
vim.g.loaded_netrwPlugin = 1

require("nvim-tree").setup()

vim.keymap.set('n', '<c-n>', ':NvimTreeFindFileToggle<CR>')
EOF
```

Create the file `init.lua`:

```
cat > init.lua <<'EOF'
require("core.plugin_config.gruvbox")
require("core.plugin_config.lualine")
require("core.plugin_config.nvim-tree")
EOF
```

Change to `~/.config/nvim` and udpate `init.lua`:

```
cd ../../.. && echo 'require("core.plugin_config")' >> init.lua
```

Open `nvim` and note the following:

1. The color scheme (configured by `gruvbox.nvim`) changed.
2. By pressing *`Ctrl+n`* you will see a file explorer (configured by `nvim-tree.lua`).
3. Note the icons (configured by `nvim-web-devicons`).
4. The status line (configured by `lualine.nvim`) now have a nice looking!

Exit `nvim`.

Do a new Git commit:

```
git add -A &&
git commit -m 'Installed and configured four (4) plugins'
```

So, until now, we have three git commits in our repository:

```
git log --oneline
```

8. Installing `treesitter` and `telescope` plugins

8.1. Installing and configuring

Add `treesitter` and `telescope`:

```
git apply <<'EOF'
diff --git a/lua/core/plugins.lua b/lua/core/plugins.lua
index bfe2bb3..d292386 100644
--- a/lua/core/plugins.lua
+++ b/lua/core/plugins.lua
@@ -17,6 +17,12 @@ return require('packer').startup(function(use)
   use 'nvim-tree/nvim-tree.lua' -- file explorer tree
   use 'nvim-tree/nvim-web-devicons' -- icons
   use 'nvim-lualine/lualine.nvim' -- status line
+  use 'nvim-treesitter/nvim-treesitter'
+  use {
+    'nvim-telescope/telescope.nvim',
+    tag = '0.1.0',
+    requires = { {'nvim-lua/plenary.nvim'} }
+  }

   -- Automatically set up your configuration after cloning packer.nvim
   -- Put this at the end after all plugins
EOF
```

Install the new plugins:

```
PackerSync
```


Buy this tutorial's NFT (details after the cover)!

Configure `telescope`:

```
cat > lua/core/plugin_config/telescope.lua <<'EOF'
local builtin = require('telescope.builtin')

vim.keymap.set('n', '<c-p>', builtin.find_files, {})
vim.keymap.set('n', '<Space><Space>', builtin.oldfiles, {})
vim.keymap.set('n', '<Space>fg', builtin.live_grep, {})
vim.keymap.set('n', '<Space>fh', builtin.help_tags, {})
EOF
```

Configure `treesitter`:

```
cat > lua/core/plugin_config/treesitter.lua <<'EOF'
require 'nvim-treesitter.configs'.setup {
  ensure_installed = {
    "vim", "lua",
    "bash",
    "java", "kotlin",
    "javascript", "typescript",
  },
  sync_install = false,
  auto_install = true,
  highlight = {
    enable = true,
  },
}
EOF
```

Add the call to those configuration files in `lua/core/plugin_config/init.lua`:

```
cat >> lua/core/plugin_config/init.lua <<'EOF'
require("core.plugin_config.telescope")
require("core.plugin_config.treesitter")
EOF
```

Open `nvim` and notice the following:

1. The `treesitter` plugin will start download the support for all the configured languages.

8.2. Adding `treesitter` support to other languages

Open `nvim` and type:

```
:TSInstallInfo
```

To install another language (like `python`), type:

```
:TSInstall python
```

8.3. Commiting our changes

```
{
  git add -A
  git commit -m 'Added treesitter and telescope plugins'
}
```

9. Adding LSP support

Buy this tutorial to read the content of this section (and the next ones). You can also **watch the video version** of this tutorial, where I execute all these sections' commands.

10. That's it for this tutorial ☺!

Again, run the command available in the section "Final Git repository".

Check that we have completed all commits and that the output is similar to the one presented in this section.

Thank you for following this tutorial. I hope this approach of copying and pasting commands directly into the terminal has been valuable for this tutorial's primary purpose and your learning Bash.