
dotfiles

Here's a bunch of settings for the various tools I use. I also have a number of blog posts and videos related to my dev environment. There's also this documentation to help you get everything installed and configured.

Documentation

- View screenshots of the current set up and how to switch themes
- Quickly get set up with these dotfiles
 - Extra WSL 1 and WSL 2 steps
- FAQ
 - How to personalize these dotfiles?
 - How to add custom themes to the set-theme script?
 - How to use a different terminal in the set-theme script?
 - How to fix Vim taking a long time to open when inside of WSL?
- About the author

Screenshots

Since my dotfiles are constantly evolving and I tend to reference them in videos, blog posts and various social media posts I thought it would be a good idea to include a screenshot of each theme I used and how to switch to it.

I prefer using themes that have good contrast ratios and are clear to see in video recordings. These dotfiles currently support easily switching between Gruvbox Community and One but you can use any theme you'd like.

Theme progression

- January 2021 (Gruvbox Community)
- April 2020 (One)
- December 2018 (Gruvbox Community)

Themes

These dotfiles include a `set-theme` script that you can run from your terminal to set your theme to any of the themes listed below. This script takes care of configuring your terminal, tmux, Vim and fzf's colors in 1 command.

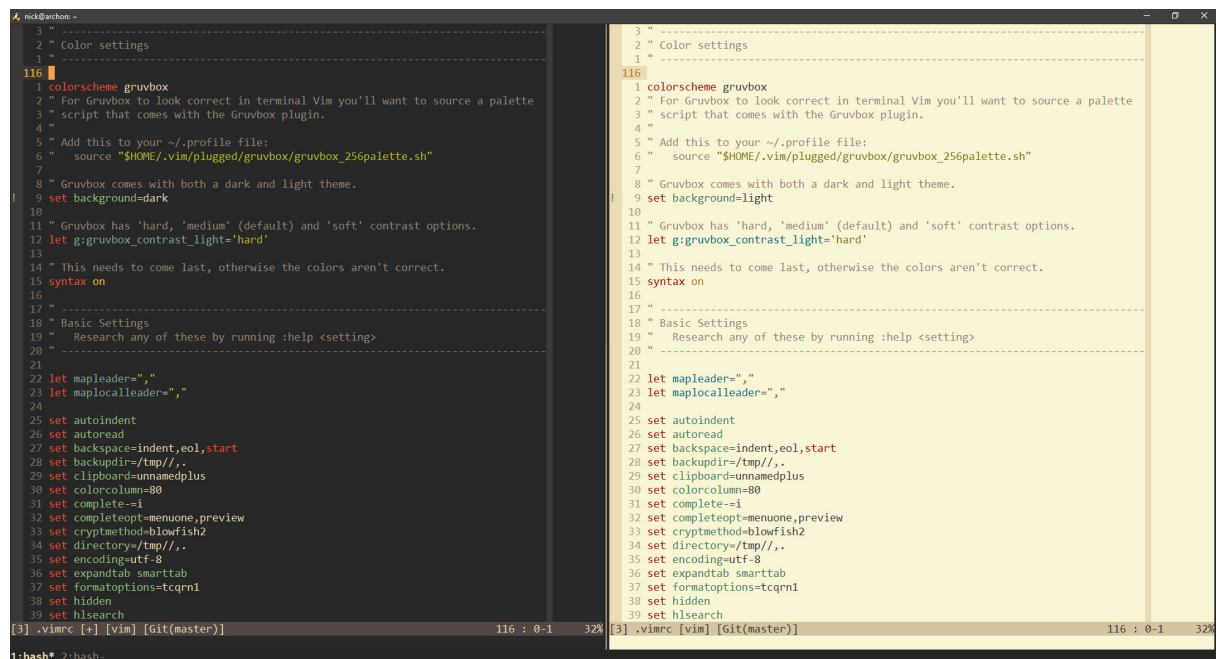
If you don't like the included themes that's no problem. You can use whatever you want, there's no limitations. You could choose to manually change the colors or adjust the `set-theme` script to add a custom theme.

Ater installing these dotfiles you can run this from your terminal:

```
1 # Switch to a supported theme.
2 # Theme names are listed below near the screenshots.
3 # You can also run set-theme --help to see a list of themes.
4 set-theme [theme_name]
5
6 # Switch between dark and light backgrounds for the active theme.
7 set-theme --toggle-bg
8
9 # Switch the theme and toggle the background in 1 command.
10 set-theme [theme_name] --toggle-bg
```

If you get an error about your terminal config file not being found please review this FAQ item.

Gruvbox Community `set-theme gruvbox`



One set-theme one

```
1 " Color settings
2 "
3 " Enable 24-bit true colors if your terminal supports it.
4 if (has("termguicolors"))
5   " https://github.com/vim/vim/issues/993#issuecomment-255651605
6   let &t_8f = "\<Esc>[38;2;%lu;%lu;%lum"
7   let &t_8b = "\<Esc>[48;2;%lu;%lu;%lum"
8
9   set termguicolors
10 endif
11
12 " Enable syntax highlighting.
13 syntax on
14
15 " Set the color scheme.
16 colorscheme one
17 set background=dark
18
19 " Status line
20 "
21 " Heavily inspired by: https://github.com/junegunn/dotfiles/blob/master/vimrc
22 function! s:statusline_expr()
23   let mod = "%{&modified ? '[+]' : !&modifiable ? '[x]' : ''}"
24   let ro = "%{&readonly ? '[RO]' : ''}"
25   let ft = "%{len(&filetype) ? '['.&filetype.']' : ''}"
26   let fug = "%{exists('g:loaded_fugitive') ? fugitive#statusline() : ''}"
27   let sep = ' '
28   let pos = '%-12(%l : %c%V%)'
29   let pct = '%p'
30
31   return '[%n] %f %<' . mod . ro . ft . fug . sep . pos . '%'.pct
32 endfunction
33
34 let &statusline = s:statusline_expr()
35 [1] .vimrc [vim] [Git(master)] 125 : 0-1 24%
```

Quickly Get Set Up with These Dotfiles

There's an `./install` script you can run to automate installing everything. That includes installing system packages such as tmux, Vim, zsh, etc. and configuring a number of tools in your home directory.

It even handles cloning down this repo. You'll get a chance to pick the clone location in the script as well as view and / or change any system packages that get installed.

The install script is optimized for:

- Ubuntu 20.04 LTS+ (native or WSL)
- Debian 11+ (Debian 10 will work if you enable backports for tmux)
- macOS 10.15+

It will still work with other distros of Linux if you skip installing system packages (more details are below).

You can download and run the install script with this 1 liner:

```
1 bash <(curl -sS https://raw.githubusercontent.com/nickjj/dotfiles/master/install)
```

If you're not comfortable blindly running a script on the internet, that's no problem. You can view the install script to see exactly what it does. Each section is commented. Sudo is only used to install system packages. Alternatively you can look around this repo and reference the config files directly without using any script.

You can also run the script without installing system packages:

```
1 bash <(curl -sS https://raw.githubusercontent.com/nickjj/dotfiles/master/install) --skip-system-packages
```

That above could be useful if you're using a non-Debian based distro of Linux, in which case you'll need to install the dependent system packages on your own beforehand. Besides that, everything else is supported since it's only dealing with files in your home directory.

My set up targets zsh 5.0+, tmux 3.0+ and Vim 8.1+. As long as you can meet those requirements you're good to go.

Did you install everything successfully?

Nice!

If you haven't done so already please close your terminal and open a new one, then follow the step(s) below:

1. Configure your git name and email One of the things the install script did was copy a git ignored git config file into your home directory. You're meant to put in your name and email address so that your details are used when you make git commits.

```
1 vim ~/.gitconfig.user
```

2. (Optional) confirm that a few things work

```
1 # Check to make sure git is configured with your name and email.
2 git config --list
3
4 # Sanity check to see if you can run some of the tools we installed.
5 vim --version
6 tmux -V
7 node --version
```

Before you start customizing certain config files, take a look at the personalization question in the FAQ.

Extra WSL 1 and WSL 2 steps

In addition to the Linux side of things, there's a few config files that I have in various directories of this dotfiles repo. These have long Windows paths.

It would be expected that you copy those over to your system while replacing "Nick" with your Windows user name if you want to use those things, such as my Microsoft Terminal `settings.json` file and others. Some of the paths may also contain unique IDs too, so adjust them as needed on your end.

Some of these configs expect that you have certain programs or tools installed on Windows. The tools I use blog post has a complete list of those tools so you can pick the ones you want to install.

Pay very close attention to the `c/Users/Nick/.wslconfig` file because it has values in there that you will very likely want to change before using it. This commit message goes into the details.

Also, you should reboot to activate your `/etc/wsl.conf` file (the install script created this). That will be necessary if you want to access your mounted drives at `/c` or `/d` instead of `/mnt/c` or `/mnt/d`.

FAQ

How to personalize these dotfiles?

Chances are you'll want to personalize some of these files, such as various Vim settings. Since this is a git repo you can always do a `git pull` to get the most up to date copy of these dotfiles, but then you may find yourself clobbering over your own personal changes.

Since we're using git here, we have a few reasonable options.

For example, from within this dotfiles git repo you can run `git checkout -b personalized` and now you are free to make whatever changes that you want on your custom branch. When it comes time to pull down future updates you can run a `git pull origin master` and then `git rebase master` to integrate any updates into your branch.

Another option is to fork this repo and use that, then periodically pull and merge updates. It's really up to you.

How to add custom themes to the set-theme script?

Prefer a video? Here's a video that demonstrates performing the steps below.

After installing these dotfiles you'll have a `~/ .local/bin/set-theme` script. It's a zero dependency Python 3 script.

1. Open the above file
2. Check out the `THEMES` dictionary near the top of the file
3. Copy one of the existing themes' dictionary items, such as `gruvbox` or `one`
4. Rename the dictionary's key to whatever your new theme's colorscheme name is in Vim
5. Adjust all of the colors and additional values in your new dictionary item as you see fit
6. Run `set-theme cooltheme`, replacing `cooltheme` with whatever name you used in step 4

Your terminal and tmux colors will update automatically, but if you have Vim already open you'll need to manually run this command from within Vim to reload your config :`so $MYVIMRC`.

If you added a theme with good contrast ratios please open a pull request to get it added to the script.

How to use a different terminal in the set-theme script?

I'm using the Microsoft Terminal but if you're using something else then your terminal's colors won't get updated by this script because the script looks for strings that are in MS terminal's config, but it's not painful to change.

By the way, if you're using the Microsoft Terminal Preview edition you'll still need to do step 1 below because the path of your MS terminal config file will be different than the non-preview edition.

You'll want to adjust the `set-theme` script by doing this:

1. Change the `TERMINAL_CONFIG` variable to reference your terminal config's path
2. Change the `terminal` attributes in the `THEMES` dictionary to use your terminal's config option names
3. Change the regex in the `change_terminal_theme` function based on your terminal's config option formatting rules
4. Optionally install Gruvbox, One or any other themes (the MS Terminal config in this repo includes them)

How to fix Vim taking a long time to open when inside of WSL?

It primarily comes down to either VcXsrv not running or a firewall tool blocking access to VcXsrv and it takes a bit of time for the connection to time out.

You can verify this by starting Vim with `vim -X` instead of `vim`. This will prevent Vim from connecting to an X server. This also means clipboard sharing to your system clipboard won't work, but it's good for a test.

Vim will try to connect to that X server by default because `DISPLAY` is exported in the `.zshrc` file. Installing and configuring VcXsrv as per these dotfiles will fix that issue.

If it still persists, it might be a software firewall issue. You can open TCP port 6000 and also restrict access to it from only WSL 2. This will depend on which tool you're using to configure that but that should do the trick.

About the Author

I'm a self taught developer and have been freelancing for the last ~20 years. You can read about everything I've learned along the way on my site at <https://nickjanetakis.com>. There's hundreds of blog posts and a couple of video courses on web development and deployment topics. I also have a podcast where I talk to folks about running web apps in production.