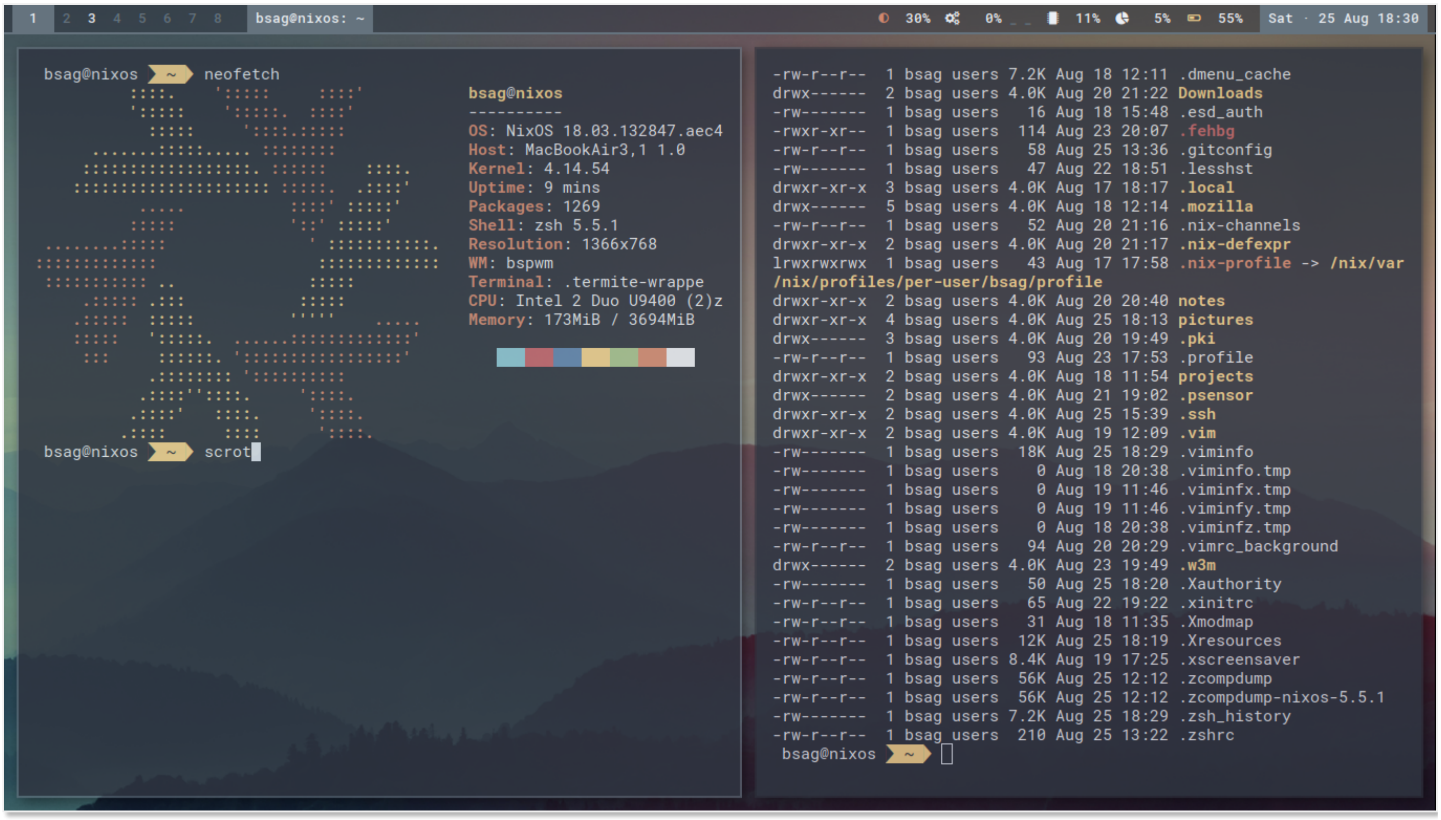
but she's a girl...

[Femina geekoides]

NixOS and the Art of OS Configuration

26 Aug 2018 • geekery



```
1  2  3  4  5  6  7  8   bsag@nixos: ~          30%   0%   11%   5%   55%   Sat · 25 Aug 18:30

bsag@nixos ~ neofetch
                                              bsag@nixos
                                              ----------
                                              OS: NixOS 18.03.132847.aec4
                                              Host: MacBookAir3,1 1.0
                                              Kernel: 4.14.54
                                              Uptime: 9 mins
                                              Packages: 1269
                                              Shell: zsh 5.5.1
                                              Resolution: 1366x768
                                              WM: bspwm
                                              Terminal: .termite-wrappe
                                              CPU: Intel 2 Duo U9400 (2)z
                                              Memory: 173MiB / 3694MiB

bsag@nixos ~ scrot
```

```
-rw-r--r--  1 bsag users 7.2K Aug 18 12:11 .dmenu_cache
drwx------  2 bsag users 4.0K Aug 20 21:22 Downloads
-rw-------  1 bsag users   16 Aug 18 15:48 .esd_auth
-rwxr-xr--  1 bsag users  114 Aug 23 20:07 .fehbg
-rw-r--r--  1 bsag users   58 Aug 25 13:36 .gitconfig
-rw-------  1 bsag users   47 Aug 22 18:51 .lesshst
drwxr-xr-x  3 bsag users 4.0K Aug 17 18:17 .local
drwx------  5 bsag users 4.0K Aug 18 12:14 .mozilla
-rw-r--r--  1 bsag users   52 Aug 20 21:16 .nix-channels
drwxr-xr-x  2 bsag users 4.0K Aug 20 21:17 .nix-defexpr
lrwxrwxrwx  1 bsag users   43 Aug 17 17:58 .nix-profile -> /nix/var
/nix/profiles/per-user/bsag/profile
drwxr-xr-x  2 bsag users 4.0K Aug 20 20:40 notes
drwxr-xr-x  4 bsag users 4.0K Aug 25 18:13 pictures
drwx------  3 bsag users 4.0K Aug 20 19:49 .pki
-rw-r--r--  1 bsag users   93 Aug 23 17:53 .profile
drwxr-xr-x  2 bsag users 4.0K Aug 18 11:54 projects
drwx------  2 bsag users 4.0K Aug 21 19:02 .psensor
drwxr-xr-x  2 bsag users 4.0K Aug 25 15:39 .ssh
drwxr-xr-x  2 bsag users 4.0K Aug 19 12:09 .vim
-rw-------  1 bsag users  18K Aug 25 18:29 .viminfo
-rw-------  1 bsag users    0 Aug 18 20:38 .viminfo.tmp
-rw-------  1 bsag users    0 Aug 19 11:46 .viminfx.tmp
-rw-------  1 bsag users    0 Aug 19 11:46 .viminfy.tmp
-rw-------  1 bsag users    0 Aug 18 20:38 .viminfz.tmp
-rw-r--r--  1 bsag users   94 Aug 20 20:29 .vimrc_background
drwx------  2 bsag users 4.0K Aug 23 19:49 .w3m
-rw-------  1 bsag users   50 Aug 25 18:20 .Xauthority
-rw-r--r--  1 bsag users   65 Aug 22 19:22 .xinitrc
-rw-r--r--  1 bsag users   31 Aug 18 11:35 .Xmodmap
-rw-r--r--  1 bsag users  12K Aug 25 18:19 .Xresources
-rw-r--r--  1 bsag users 8.4K Aug 19 17:25 .xscreensaver
-rw-r--r--  1 bsag users  56K Aug 25 12:12 .zcompdump
-rw-r--r--  1 bsag users  56K Aug 25 12:12 .zcompdump-nixos-5.5.1
-rw-------  1 bsag users 7.2K Aug 25 18:29 .zsh_history
-rw-r--r--  1 bsag users  210 Aug 25 13:22 .zshrc
 bsag@nixos ~
```

You know how it is when you take a quick look at a framework for organising your dotfiles, and end up installing and configuring a new Linux operating system on an old MacBook Air? No? Just me then.

It started innocuously enough. I've been really busy at work all summer, with lots of travelling, a lot of working weekends, and long hours. That hasn't really left enough blocks of time for sewing projects, so I've been falling back on my other relaxation activity: pottering about with computers in odd moments. I had seen NixOS before, but came across it again because it acts as a package manager and also potentially a way of managing your configuration files.

Home Manager, which uses the Nix package manager component to both install and configure applications on your system, is intriguing. All of the Nix elements use the Nix language, which is a purely functional language. This means that you can set up your configuration to be completely declarative, defining which applications you will install (what version, which plugins you will use and so on), as well as your own personal configuration files (like the contents of your `.zshrc`), all in one file. Then when you execute that file, your system is set to the state you have declared in your configuration, with no broken dependencies. This raises the interesting possibility of being able to bootstrap an entire system easily from scratch, once you have got the configuration right.

Home Manager is built on Nix (the package manager), which in turn is a component of NixOS, the Linux operating system. You can install Nix on a variety of platforms (including macOS), but the operating system provides a self-contained Linux-based system. Since I was finding it a bit difficult to figure out how Home Manager worked without knowing about Nix, I decided to install the package manager on my Mac.

Once you've got it installed, it works in a very similar way to other package managers like Homebrew. You issue a command to install a particular bit of software, and it does so. The interesting part is that it stores all the installed software in a `/nix/store` directory, with links so that the system can find binaries, libraries and so on. Even better, you can safely install multiple versions of the same software without getting in a mess, uninstall software and its particular dependencies cleanly, and you can even roll back the system to a previous state if you mess something up. There's also a `nix-shell` tool, which enables you to specify a particular set of packages inside a project directory. So you could set up a coding project with a particular version of Python, a set of pip packages and so on, all of which is activated when you move into that directory, but none of which affects (or is affected by) whatever else you have installed on the system. Neat. That may sound very like Python's virtual environments, or Ruby's Bundle system, but it is language-independent, so you could handle all your project dependencies that way, irrespective of the actual tools used.

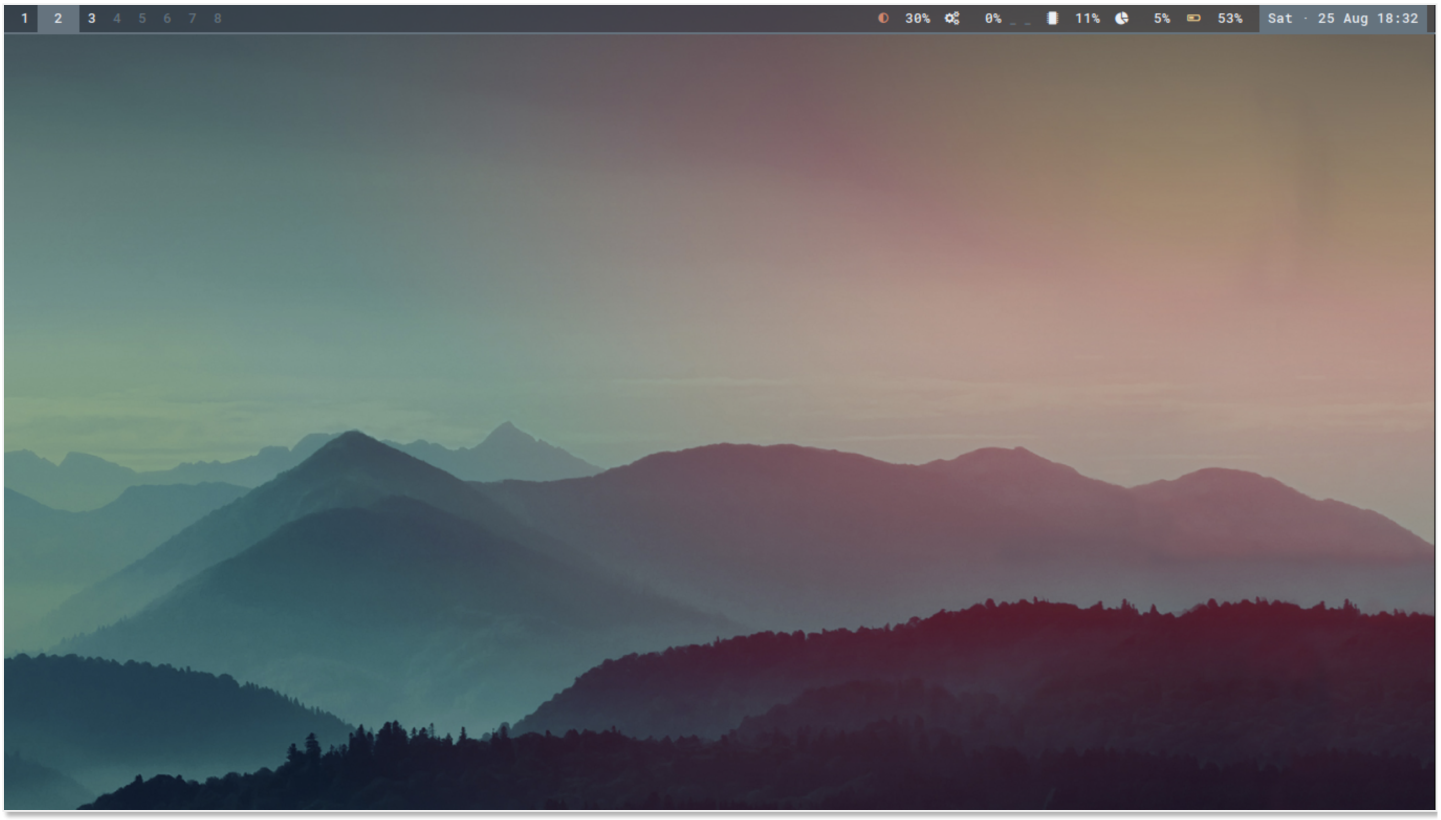
The more I read about Nix, the more interesting it became, and I got the crazy idea that I could try installing NixOS. The operating system has an even closer tie to the Nix ecosystem, as you might expect, so you can configure your whole operating system (hardware, software, everything) from a couple of configuration files, and then reproduce that completely reliably. It has the same rollback feature, so again, if you mess something up, you can just revert back to the last version and try again. I started off installing NixOS on my Mac via Virtualbox, and set up a basic system. It worked so smoothly, that I got the confidence to install it on an 8 year old MacBook Air that I had sitting around and not doing much. And so it begins...

Installing any version of Linux on a Mac is always a little bit of an adventure. However, I knew from this guide written by Bernerd Schaefer that it was possible. I ended up diverging from that guide though. The main difficulty I encountered was with partitioning and formatting the drive correctly. I could boot from the NixOS installation image that I put on a USB stick, and also generate the `hardware.nix` and `configuration.nix` files with the `nixos-generate-config` command, but then it would fail to boot with errors reported in `hardware.nix`.

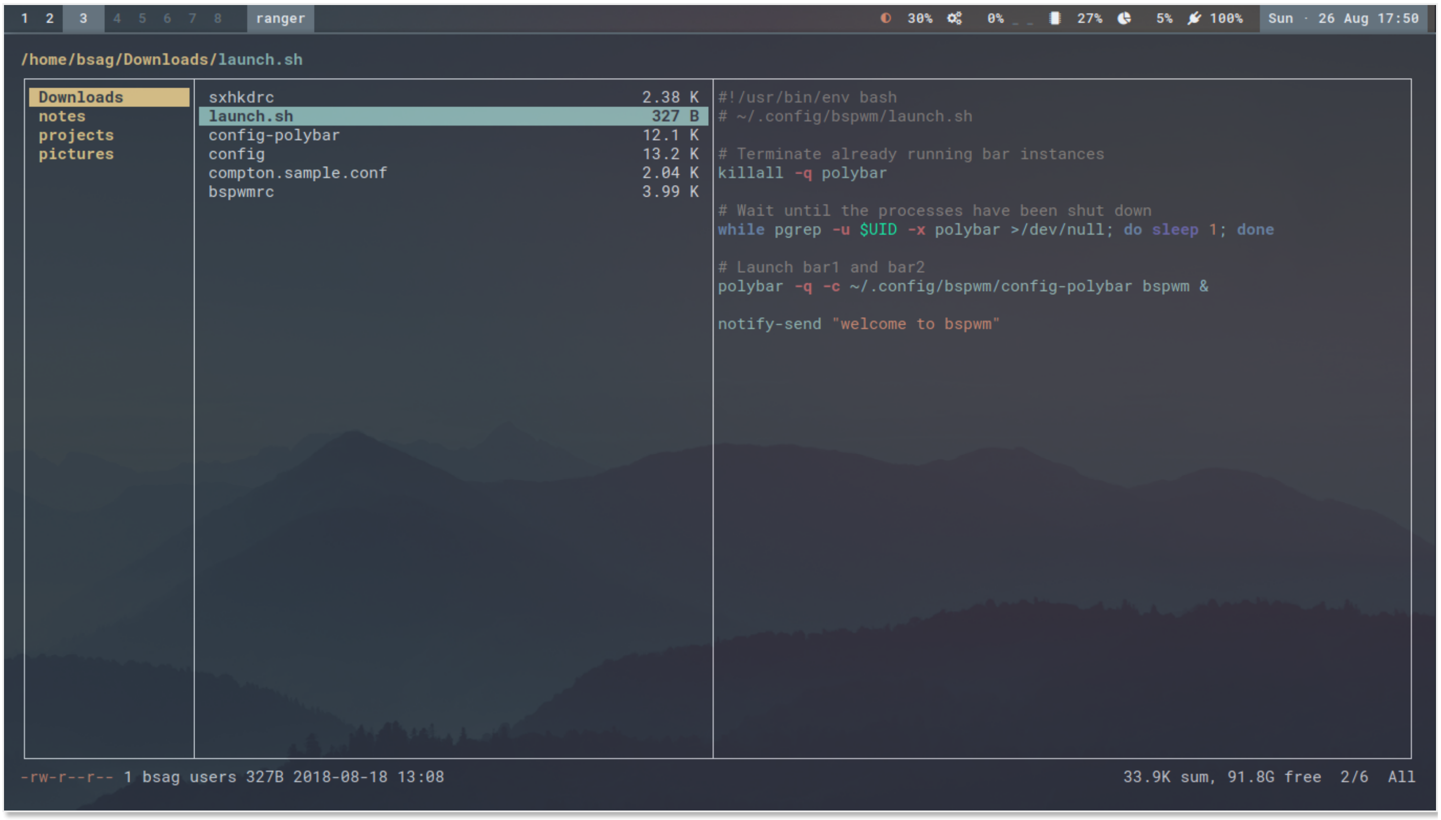
I abandoned the idea of using a LUKS encrypted volume for simplicity, and did a lot of searching before I found the source of the problem. Essentially, if you format a single partition in macOS with a GUID partition table (called GPT in Linux), it creates a hybrid GPT/MBR partition table (apparently to allow dual booting with older versions of Windows). What you need to do once you boot from the NixOS installation stick is use `fdisk` to delete the main partition (leaving the EFI boot partition alone), and create a new one, formatting it as Linux filesystem. Then you should have the correct EFI boot setup, which is GPT with protected MBR. I have only the haziest idea what that all means, but that was what did the trick. Then I was able to use `fdisk` again to create the following partitions (where `/dev/sda` is my internal SSD):

- Partition 1 (200M): this is the EFI boot partition which was created when I partitioned the disk originally in macOS. Don't touch this! It should be labelled 'EFI System', and the other partitions you create should be appended to the sectors after the end of this partition.
- Partition 2 (512M): the boot partition (EFI System). You need to create this in addition to Partition 1. This initially confused me, because I thought Partition 1 would do the job.
- Partition 3 (8G in my case): swap partition, Linux filesystem. You could manage with a much smaller partition, but I have only 4G of RAM on this machine, so I went with a generous amount of swap.
- Partition 4 (the rest of the space): root partition, Linux root (x86-64) format.

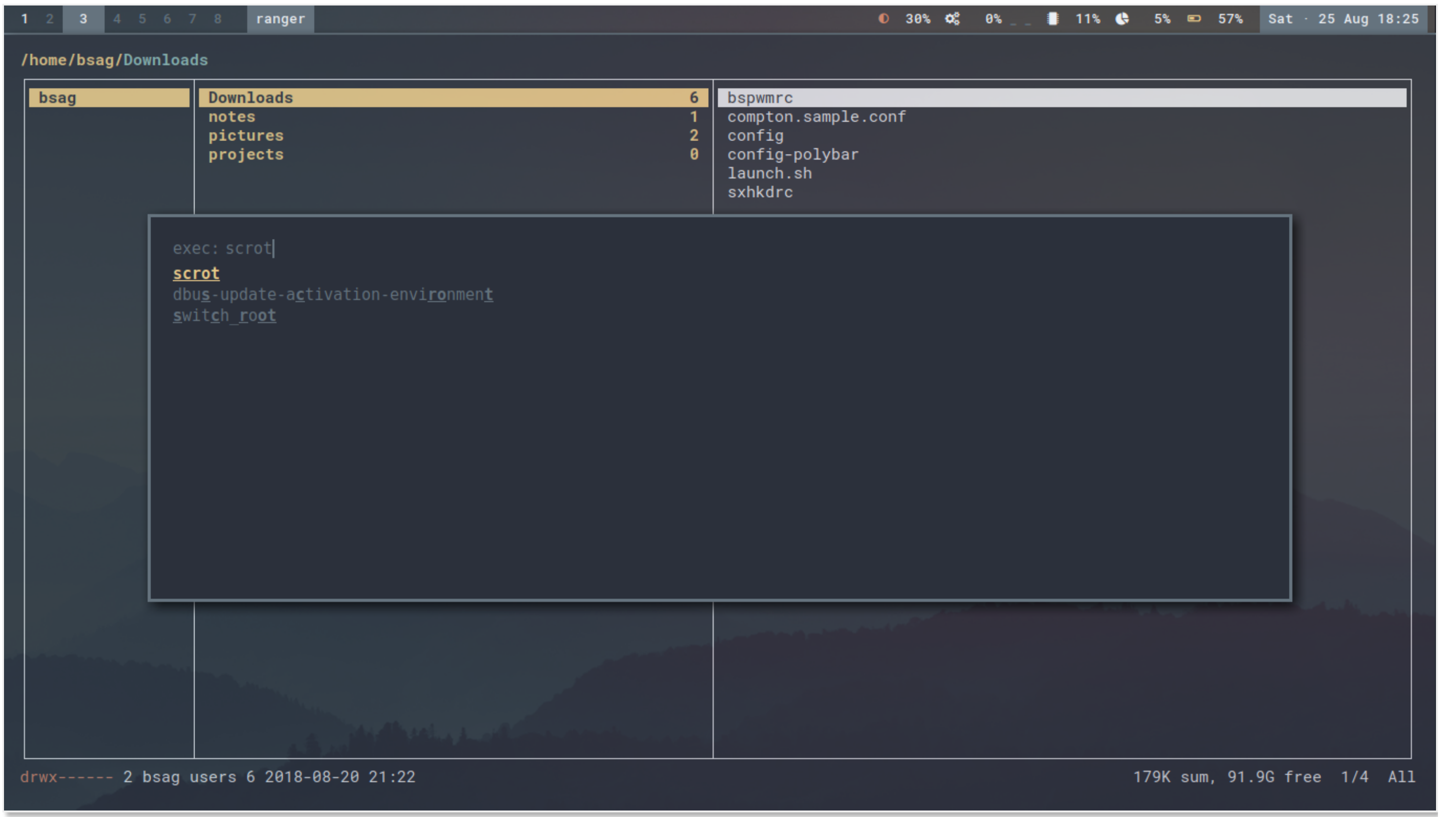
That should do the trick. If you generate the Nix config again, you should see that `hardware.nix` has captured your filesystem correctly, and it should boot without problems. Once I got to this stage, everything was much easier. When you have rebooted, you can start editing `configuration.nix` to add the packages you want, and even define users declaratively (you can see my configuration.nix here).



I have been very interested in minimal, tiling window managers for a while, so I started with a configuration developed by Brihadeesh here, using Bastien Dejean's bspwm as the window manager, with sxhkd as a hotkey manager. When Brihadeesh posted his setup on Reddit, he used a wallpaper by AidenDrew that I fell in love with, so I use that too. It goes very nicely with the Nord colour scheme that I'm fond of. I use the Compton compositor to give windows transparency and shadow, and Rofi to launch applications and switch windows. I must say that a tiling window manager is great on such a small screen, and really makes the best use of the screen real estate. It's also a nicely consistent system, both visually (I can use the Nord colourscheme pretty much everywhere), and in terms of keyboard commands (everything uses Vim-inspired keys).

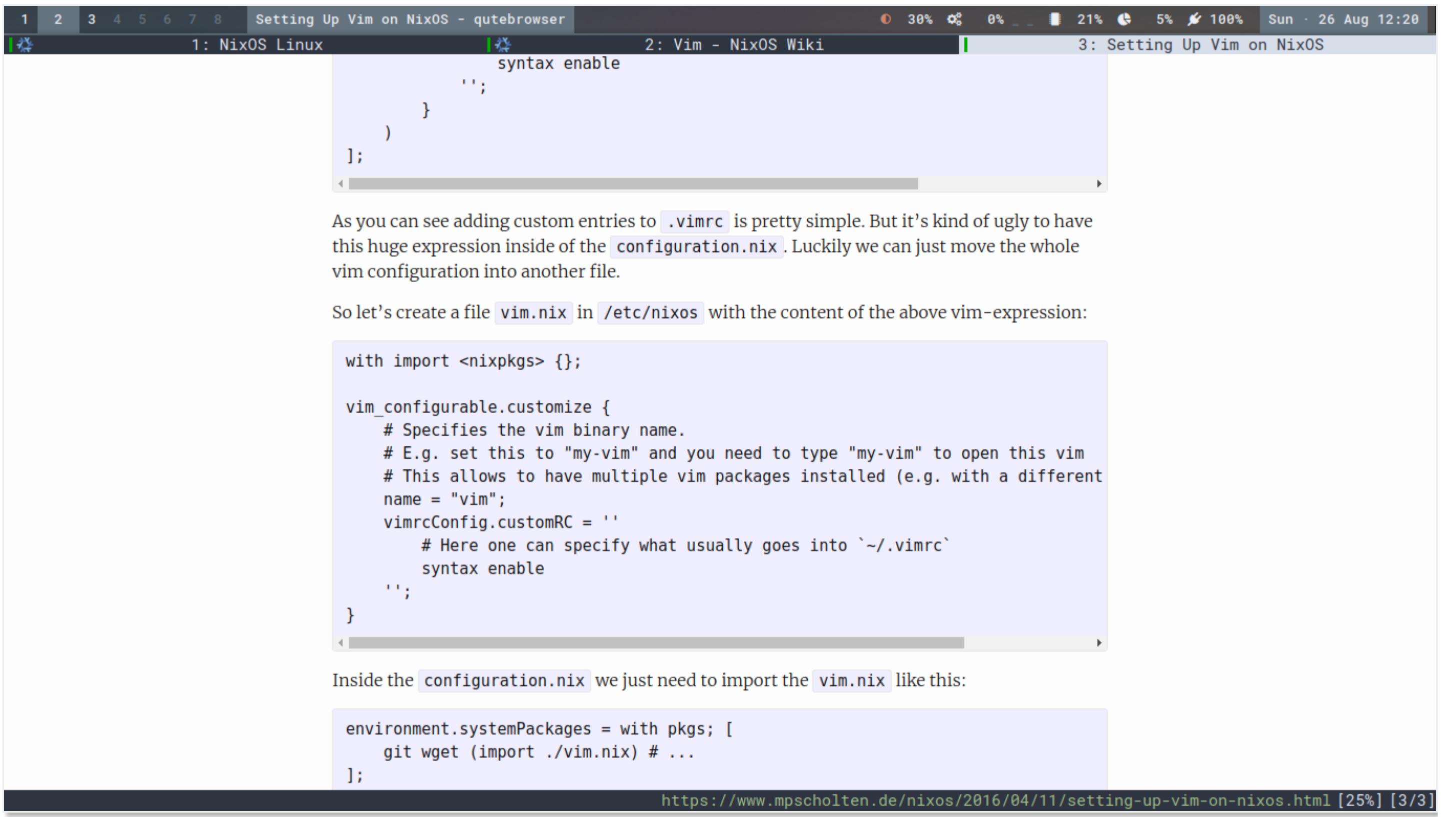


```
/home/bsag/Downloads/launch.sh

Downloads   sxhkdrc               2.38 K  #!/usr/bin/env bash
notes       launch.sh              327 B  # ~/.config/bspwm/launch.sh
projects    config-polybar        12.1 K
pictures    config                13.2 K  # Terminate already running bar instances
            compton.sample.conf   2.04 K  killall -q polybar
            bspwmrc               3.99 K
                                          # Wait until the processes have been shut down
                                          while pgrep -u $UID -x polybar >/dev/null; do sleep 1; done

                                          # Launch bar1 and bar2
                                          polybar -q -c ~/.config/bspwm/config-polybar bspwm &

                                          notify-send "welcome to bspwm"

-rw-r--r-- 1 bsag users 327B 2018-08-18 13:08                    33.9K sum, 91.8G free  2/6  All
```



```
1  2  3  4  5  6  7  8   ranger                    30%    0%    11%    5%    57%   Sat · 25 Aug 18:25

/home/bsag/Downloads

bsag     Downloads       6   bspwmrc
         notes           1   compton.sample.conf
         pictures        2   config
         projects        0   config-polybar
                             launch.sh
                             sxhkdrc

     exec: scrot
     scrot
     dbus-update-activation-environment
     switch_root

drwx------ 2 bsag users 6 2018-08-20 21:22                       179K sum, 91.9G free  1/4  All
```

Most of the other packages I have installed are similarly minimal or commandline only. I use Ranger as a file manager, the glorious Zathura PDF viewer, and along the way I discovered Qutebrowser. Qutebrowser is fantastic, with embedded Vim-inspired hotkeys for moving around tabs, history and even clicking links, all from the keyboard. It also has a very minimal UI, which works very well with a tiling window manager, and makes the most of the tiny 11" MacBook Air screen. As far as the hardware goes, I've got the trackpad working properly, and I can use screen brightness and volume keys to change those settings. I also discovered (more internet searching) that the package `tlp` offers much improved power management, so the Air is running much cooler now and gets around 4 hours of battery life, which isn't bad considering it is 8 years old and the battery is not in the best condition.



```
1  2  3  4  5  6  7  8   Setting Up Vim on NixOS - qutebrowser      30%    0%    21%    5%   100%   Sun · 26 Aug 12:20
        1: NixOS Linux             2: Vim - NixOS Wiki             3: Setting Up Vim on NixOS
```

```
                syntax enable
              '';
          }
      )
  ];
```

As you can see adding custom entries to `.vimrc` is pretty simple. But it's kind of ugly to have this huge expression inside of the `configuration.nix`. Luckily we can just move the whole vim configuration into another file.

So let's create a file `vim.nix` in `/etc/nixos` with the content of the above vim-expression:

```
with import <nixpkgs> {};

vim_configurable.customize {
    # Specifies the vim binary name.
    # E.g. set this to "my-vim" and you need to type "my-vim" to open this vim
    # This allows to have multiple vim packages installed (e.g. with a different
    name = "vim";
    vimrcConfig.customRC = ''
        # Here one can specify what usually goes into `~/.vimrc`
        syntax enable
    '';
}
```

Inside the `configuration.nix` we just need to import the `vim.nix` like this:

```
environment.systemPackages = with pkgs; [
    git wget (import ./vim.nix) # ...
];
```

```
https://www.mpscholten.de/nixos/2016/04/11/setting-up-vim-on-nixos.html [25%] [3/3]
```

I'm really pleased with this meander into Linux World. I haven't finished yet, though. I've realised that I probably do need to use Home Manager to make it easier to manage all the dotfiles and other configuration files alongside the main system configuration, which brings me right back where I started. *Hakuna matata*.

I'm really impressed with NixOS. It certainly isn't the easiest Linux to install, and you have to do a lot of searching around outside the manual to find some of the answers you need, but it is a very robust and sleek system. Despite the fact that I made no end of errors, I never ended up in a situation that I could not recover from. When you edit `configuration.nix` and use it to rebuild the system, the compiler picks up on syntax errors so you can fix them before trying again. And if the worst does happen and the compiler misses some critical thing resulting in a broken system, you can reboot, and use the boot menu to pick the previous working generation of your system. That makes it a fun and worry-free system to experiment with and learn about building a system. I'm keen to see if I can use Home Manager on my Mac to replace Homebrew and build a system with which I can bootstrap a new installation to my liking.

← OLDER NEWER →

Login

Add a comment

M↓ MARKDOWN COMMENT ANONYMOUSLY ADD COMMENT

Powered by **Commento**

Search...

Activity elsewhere

Social media