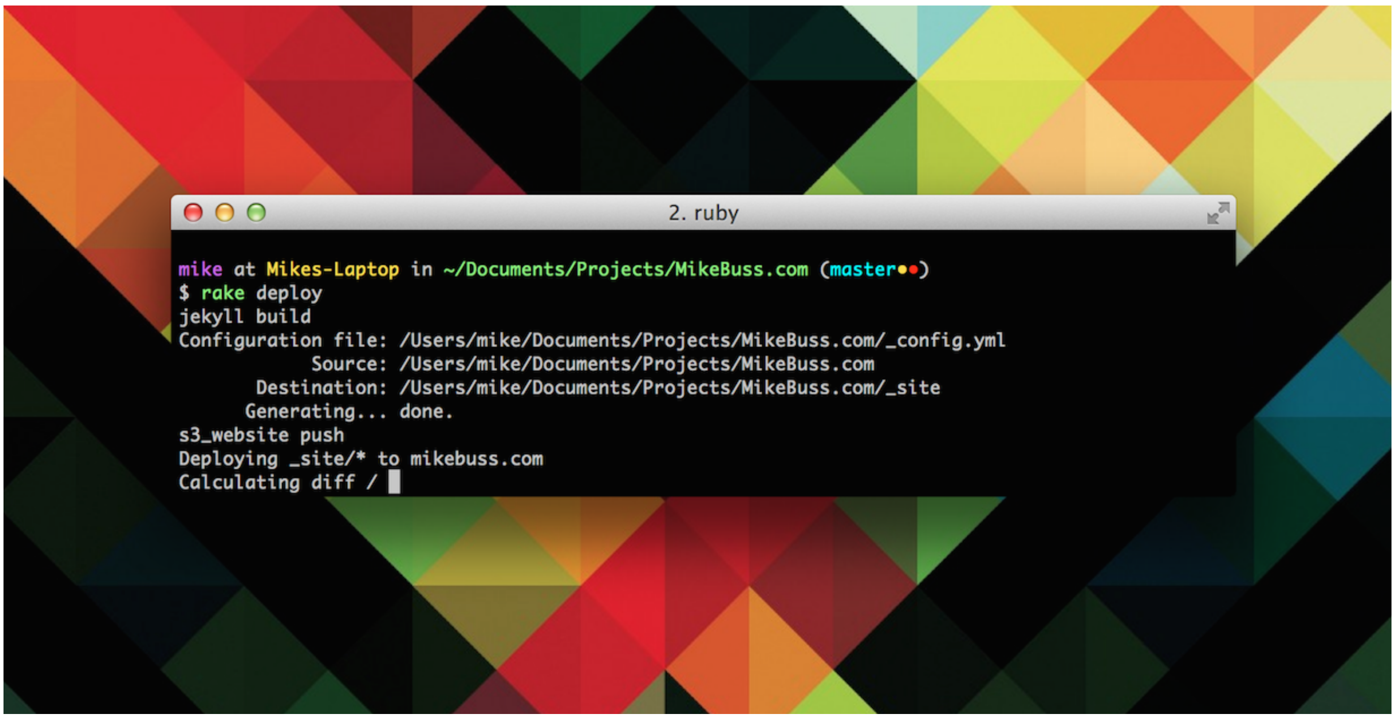
A Beautifully Productive Terminal Experience

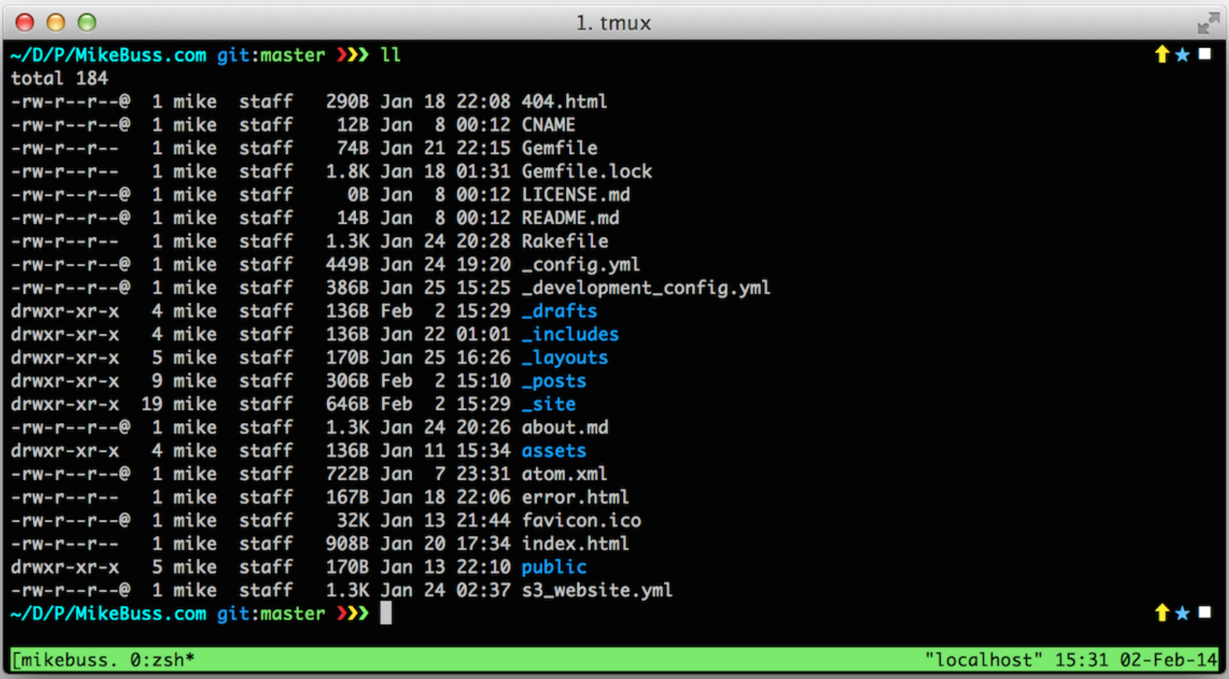


```
mike at Mikes-Laptop in ~/Documents/Projects/MikeBuss.com (master●●)
$ rake deploy
jekyll build
Configuration file: /Users/mike/Documents/Projects/MikeBuss.com/_config.yml
            Source: /Users/mike/Documents/Projects/MikeBuss.com
       Destination: /Users/mike/Documents/Projects/MikeBuss.com/_site
      Generating... done.
s3_website push
Deploying _site/* to mikebuss.com
Calculating diff /
```

Every developer can benefit from using the command line. Whether you use it to bounce between projects or create complex scripts, getting familiar with the command line can drastically improve your workflow.

In this post, we'll cover some of the great tools available for streamlining your command line experience.

After some experimenting, I've found that using a combination of iTerm 2, Zsh, Prezto, Tmux, and Tmuxinator makes for an extremely productive setup. I'll go over each of these tools and then show how they can be combined to improve your development workflow.



```
~/D/P/MikeBuss.com git:master >>> ll
total 184
-rw-r--r--@  1 mike  staff   290B Jan 18 22:08 404.html
-rw-r--r--@  1 mike  staff    12B Jan  8 00:12 CNAME
-rw-r--r--   1 mike  staff    74B Jan 21 22:15 Gemfile
-rw-r--r--   1 mike  staff   1.8K Jan 18 01:31 Gemfile.lock
-rw-r--r--@  1 mike  staff     0B Jan  8 00:12 LICENSE.md
-rw-r--r--@  1 mike  staff    14B Jan  8 00:12 README.md
-rw-r--r--   1 mike  staff   1.3K Jan 24 20:28 Rakefile
-rw-r--r--@  1 mike  staff   449B Jan 24 19:20 _config.yml
-rw-r--r--@  1 mike  staff   386B Jan 25 15:25 _development_config.yml
drwxr-xr-x   4 mike  staff   136B Feb  2 15:29 _drafts
drwxr-xr-x   4 mike  staff   136B Jan 22 01:01 _includes
drwxr-xr-x   5 mike  staff   170B Jan 25 16:26 _layouts
drwxr-xr-x   9 mike  staff   306B Feb  2 15:10 _posts
drwxr-xr-x  19 mike  staff   646B Feb  2 15:29 _site
-rw-r--r--@  1 mike  staff   1.3K Jan 24 20:26 about.md
drwxr-xr-x   4 mike  staff   136B Jan 11 15:34 assets
-rw-r--r--@  1 mike  staff   722B Jan  7 23:31 atom.xml
-rw-r--r--   1 mike  staff   167B Jan 18 22:06 error.html
-rw-r--r--@  1 mike  staff    32K Jan 13 21:44 favicon.ico
-rw-r--r--   1 mike  staff   908B Jan 20 17:34 index.html
drwxr-xr-x   5 mike  staff   170B Jan 13 22:10 public
-rw-r--r--@  1 mike  staff   1.3K Jan 24 02:37 s3_website.yml
~/D/P/MikeBuss.com git:master >>> 

[mikebuss. 0:zsh*                                                    "localhost" 15:31 02-Feb-14
```

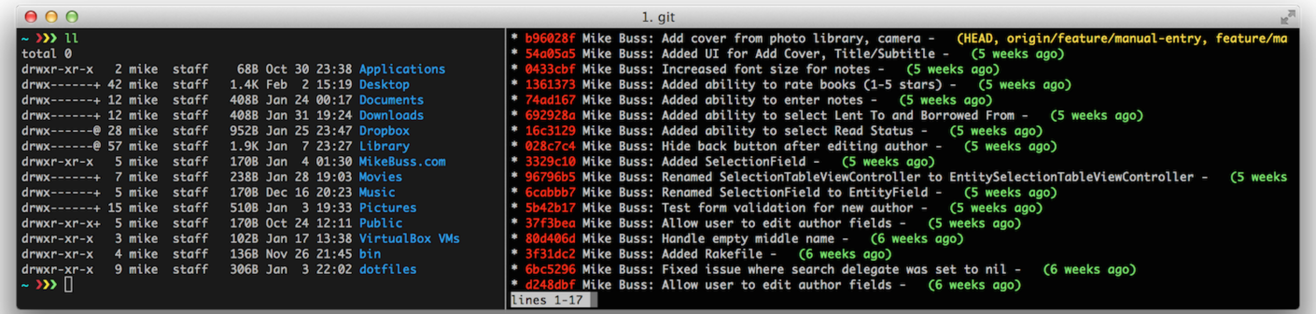
Note: It should go without saying that there's no *right* way to work. I prefer to use the command line because I'm a fast typist and hate using the mouse, but everyone has their own opinion. Let me know how you prefer to work!

---

iTerm 2

iTerm 2 is a replacement for the Terminal app that ships with Mac OS X. I use it for these great features:

- Split pane views
- Hotkey Window that gives you an always-available Terminal window
- Better search highlighting
- Mouseless copy



```
~ >>> ll
total 0
drwxr-xr-x   2 mike  staff    68B Oct 30 23:38 Applications
drwx------+ 42 mike  staff   1.4K Feb  2 15:19 Desktop
drwx------+ 12 mike  staff   408B Jan 24 00:17 Documents
drwx------+ 12 mike  staff   408B Jan 31 19:24 Downloads
drwx------@ 28 mike  staff   952B Jan 25 23:47 Dropbox
drwx------@ 57 mike  staff   1.9K Jan  7 23:27 Library
drwxr-xr-x   5 mike  staff   170B Jan  4 01:30 MikeBuss.com
drwx------+  7 mike  staff   238B Jan 28 19:03 Movies
drwx------+  5 mike  staff   170B Dec 16 20:23 Music
drwx------+ 15 mike  staff   510B Jan  3 19:33 Pictures
drwxr-xr-x+  5 mike  staff   170B Oct 24 12:11 Public
drwxr-xr-x   3 mike  staff   102B Jan 17 13:38 VirtualBox VMs
drwxr-xr-x   4 mike  staff   136B Nov 26 21:45 bin
drwxr-xr-x   9 mike  staff   306B Jan  3 22:02 dotfiles
~ >>> ▯
```

```
* b96028f Mike Buss: Add cover from photo library, camera -   (HEAD, origin/feature/manual-entry, feature/ma
* 54a05a5 Mike Buss: Added UI for Add Cover, Title/Subtitle -   (5 weeks ago)
* 0433cbf Mike Buss: Increased font size for notes -   (5 weeks ago)
* 1361373 Mike Buss: Added ability to rate books (1-5 stars) -   (5 weeks ago)
* 74ad167 Mike Buss: Added ability to enter notes -   (5 weeks ago)
* 692928a Mike Buss: Added ability to select Lent To and Borrowed From -   (5 weeks ago)
* 16c3129 Mike Buss: Added ability to select Read Status -   (5 weeks ago)
* 028c7c4 Mike Buss: Hide back button after editing author -   (5 weeks ago)
* 3329c10 Mike Buss: Added SelectionField -   (5 weeks ago)
* 96796b5 Mike Buss: Renamed SelectionTableViewController to EntitySelectionTableViewController -   (5 weeks
* 6cabbb7 Mike Buss: Renamed SelectionField to EntityField -   (5 weeks ago)
* 5b42b17 Mike Buss: Test form validation for new author -   (5 weeks ago)
* 37f3bea Mike Buss: Allow user to edit author fields -   (5 weeks ago)
* 80d406d Mike Buss: Handle empty middle name -   (6 weeks ago)
* 3f31dc2 Mike Buss: Added Rakefile -   (6 weeks ago)
* 6bc5296 Mike Buss: Fixed issue where search delegate was set to nil -   (6 weeks ago)
* d248dbf Mike Buss: Allow user to edit author fields -   (6 weeks ago)
lines 1-17
```

You can find a complete list of features here.

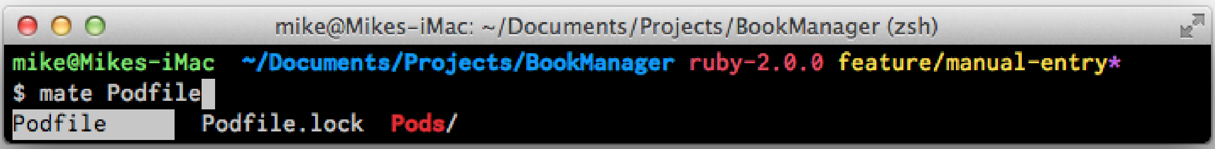
ZSH Over Bash

ZShell (Zsh) is a shell designed for interactive use, and can be used as a powerful command interpreter for shell scripting.

There are very few differences between Bash, the default shell for Mac OS X, and Zsh for the average user. I chose to switch to Zsh because it has some fantastic community-driven frameworks available.

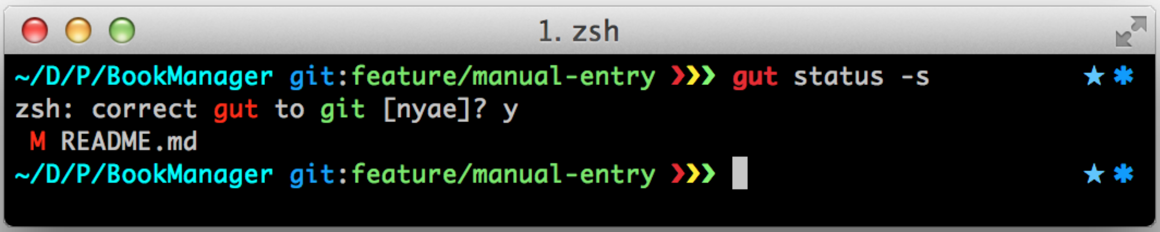
Here are some of the advantages of Zsh when compared to Bash:

- **Better completion handling.** Not only are completions command-specific, Zsh also adds the ability to navigate the completion list by hitting `<Tab>`.



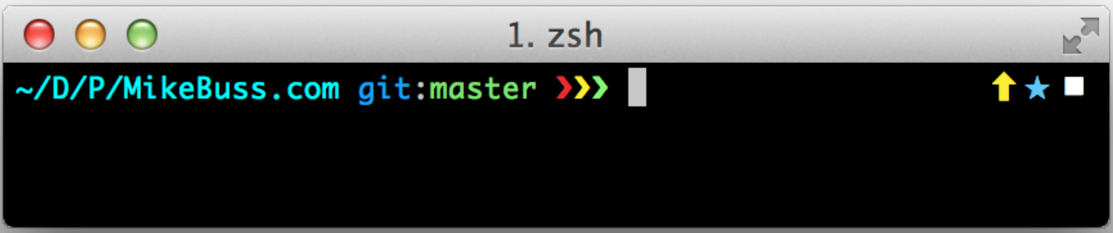
```
mike@Mikes-iMac  ~/Documents/Projects/BookManager ruby-2.0.0 feature/manual-entry*
$ mate Podfile
Podfile       Podfile.lock   Pods/
```

- **Spelling correction.** If you spell something wrong, Zsh can sometimes suggest a correction and re-run the command for you.



```
1. zsh
~/D/P/BookManager git:feature/manual-entry ❯❯❯ gut status -s                  ★ ✱
zsh: correct gut to git [nyae]? y
 M README.md
~/D/P/BookManager git:feature/manual-entry ❯❯❯                                 ★ ✱
```

- **Right-hand prompts.** Zsh gives you the ability to add supplementary information to the right side of your prompt. I use this to display git information, but I've seen others use it for battery indicators.



```
1. zsh
~/D/P/MikeBuss.com git:master ❯❯❯                    ⬆ ★ ◼
```

Let's take a look at two of the most popular Zsh frameworks: **oh-my-zsh** and **prezto**.

OH MY ZSHELL!

Oh-my-zsh is the first framework we'll look at. With over 120 optional plugins and 120 themes, oh-my-zsh is the largest Zsh framework. After using it for several years, I've found it to be very stable and easy to use.

Note: I've since switched to Prezto, but oh-my-zsh is a fantastic framework worth mentioning. Read more about how to decide between the two frameworks by scrolling down to the Choosing Between Oh-my-zsh and Prezto section.

Plugins

The real power behind these frameworks comes from their plugins. When I was using oh-my-zsh, here were the plugins I used the most:

- git - A great plugin that adds a bunch of fantastic aliases. This was my most used plugin when I was using oh-my-zsh.
- git flow - I use the Git Flow branching model for git. The `git flow` plugin adds a bunch of git extensions that take a few steps out of the process.
- github - A plugin that adds auto-complete to the github gem, which is super useful for interacting with GitHub from the command line. (hub is also a great option for this)
- brew - A great auto-completion plugin for HomeBrew, *the* package manager for Mac OS X.
- bundler - Occasionally I'll have a Gemfile with my iOS projects to help manage any helper Ruby scripts I have, in addition to locking down a project to a specific version of CocoaPods. This plugin adds a few shortcuts for Bundler, including exec, update, and install.
- osx - A plugin that adds a few aliases that interact with Finder.
- textmate - A bunch of helper aliases for opening files and directories in my favorite editor, Textmate.

- sublime - For those who prefer Sublime Text over Textmate, this plugin is great for opening files and directories in Sublime.
- tmux - A plugin that adds several options for effecting the startup behavior of tmux, my terminal multiplexer of choice.
- tmuxinator - A plugin that adds completions for Tmuxinator, a tool I'll be covering later in this post.

A full list of plugins can be found here.

If you'd like to go ahead and install oh-my-zsh, the official instructions can be found on their GitHub repository. Before you do, though, you should see if Prezto is better suited for your needs.

Prezto

From the official Prezto repository:

> *Prezto is the configuration framework for Zsh; it enriches the command line interface environment with sane defaults, aliases, functions, auto completion, and prompt themes.*

Prezto is a fork of oh-my-zsh that has been rewritten for speed. After using it for the past several months, I've found it's much faster than my oh-my-zsh setup.

Prezto also has themes and a fantastic selection of plugins (although Prezto calls them modules). Here are the modules I use the most:

- git - Adds some great aliases for git.
- utility - Adds some great general-purpose aliases.
- homebrew - Adds useful HomeBrew aliases.
- ruby - Adds aliases for working with Ruby gems, RVM, rbenv, and Bundler.
- rails - Adds some great aliases for working with Rails.

- tmux - Adds support for starting tmux automatically at startup and some great aliases.
- syntax-highlighting - Adds syntax highlighting for commands to zsh. Make sure to load this module second to last, where the last module is prompt.
- osx - Adds aliases for Mac OS X, including support for QuickLook.
- history - Adds history options and aliases. Running `history-stat` gives you a cool list of your top 10 most used commands.
- wakeonlan - Adds support for waking computers using the wakeonlan tool.

The cover image for this post uses the steeef theme for Prezto. More info on customizing your Prezto prompt here!

Choosing Between Oh-my-zsh and Prezto

Oh-my-zsh has a much larger selection of themes and plugins, but Prezto's tab completion is noticeably faster.

If you have a plugin you'd really miss by switching to Prezto, I'd suggest sticking with oh-my-zsh. Otherwise, the speed increase is well worth the switch.

Tmux

Tmux is a terminal multiplexer. Terminal multiplexers lets you easily switch between several programs in one terminal, which makes them the perfect tool for moving from one workspace to another.

After using tmux for over a year, I can't imagine life without it. After memorizing a few keystrokes, it's extremely easy to switch between projects.

The website Giant Robots Smashing Into Other Giant Robots has a great crash course on tmux. It's definitely worth a read!

Tmux can be installed with HomeBrew using `brew install tmux`.

Tmuxinator

Tmuxinator is a Ruby gem that makes creating and managing complex tmux sessions a breeze.

Tmuxinator can be installed with `gem install tmuxinator`.

Feeling Social? Try Wemux

Wemux adds powerful, easy to use multi-user terminal multiplexing. If you've ever wanted to share your terminal with someone else, this is a great way to do it. It has several modes that allow for different access privileges.

Wemux can also be installed using HomeBrew using `brew install wemux`.

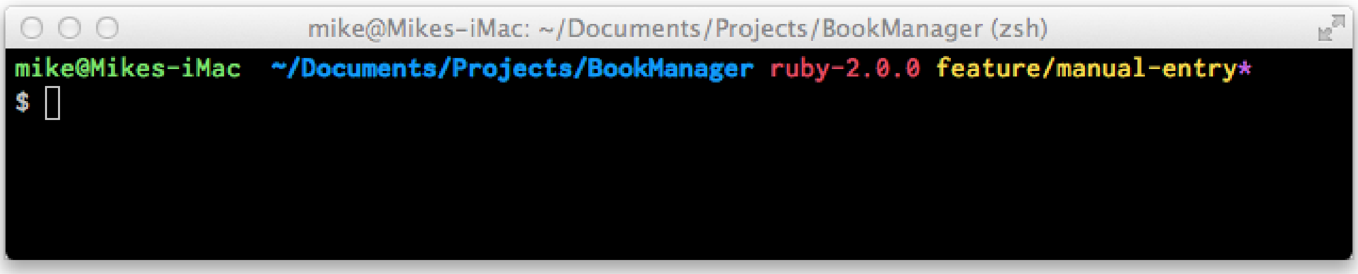
Putting It All Together

So you've got all these great tools installed. Now what? Let's go over how you can work them into your workflow.

Git Integration

Both frameworks have themes that let you put git information in your prompt.

Here's an example of a great oh-my-zsh prompt:



```
mike@Mikes-iMac  ~/Documents/Projects/BookManager ruby-2.0.0 feature/manual-entry*
$ 
```

Just from looking at the prompt, you can tell:

- Current machine name
- Current working directory
- Current Ruby version (using RVM)
- Current git branch
- Whether there are uncommitted changes (the asterisk)

Having all of this information available every time you enter your project's directory is invaluable. The less typing you have to do, the better!

Long-Running Processes

There are some **great** command line tools out there for iOS developers. Nomad-CLI is a great example, with 5 tools all geared towards iOS development and testing.

Let's go over one way to work with long-running processes using tmux. First, make sure you have tmux installed. Using HomeBrew, it's as simple as running:

```
$ brew install tmux
```

Next, let’s install Shenzhen, a tool for creating and distributing an IPA for your iOS app.

```
$ gem install shenzhen
```

Now that we have both tools installed, go ahead and start a new tmux session called ‘deploy’:

```
$ tmux new-session -s deploy
```

Navigate to your iOS project’s directory and build your project using this command:

```
$ ipa build
```

If this works, you should see your IPA file in the same directory. Now for the fun part!

You can distribute your app through TestFlight, HockeyApp, or FTP using Shenzhen, so let’s give that a try. If you have one of these distribution methods, feel free to follow along. If not, just take my word that this command works!

```
$ ipa distribute:sftp --host HOST -u USER -p PASSWORD -P
FTP_PATH
```

Depending on the size of your app and your connection, uploading the IPA can take a considerable amount of time. This is where tmux comes in.

While the IPA is uploading, try detaching your session by hitting `CTRL+b` and then `d` with the window in focus. You should return to the prompt you saw before running the `tmux new-session` command.

If we’re back to the original prompt, where did the upload go? Well, detached tmux sessions continue to run in the background. Let’s go back to our long-running upload process by reattaching the session:

```
tmux attach-session -t deploy
```

And voilà! You’re back to the upload process.

Tmux is a great way to switch back and forth between workspaces. Using it to put a process in the background and check back on it’s process periodically is just one small example of what tmux is capable of - this example is barely scratching the tip of the iceberg.

Note: Checking back on a long-running process is fine, but if you’d like to receive a push notification when it finishes, check out Push Notifications from the Command Line Using Pushover. The tutorial works even better when you incorporate tmux into the mix.

Conclusion

Getting familiar with the command line can drastically improve your workflow. If you prefer using the keyboard over the mouse, you should try out some of these great tools.

Have a suggestion for a tool? Want to share your workflow? Feel free to send me a line!

More Information

- Zsh is your friend
- Hey Prezto - zsh for command line heaven
- My extravagant Zsh Prompt

- Ditching Oh-my-zsh for Prezto
- Why Zsh is cooler than your shell
- A tmux Crash Course
- My new ZSH prompt
- Using Tmux and Tmuxinator
- Beautiful Tools

Mike Buss

I'm a software engineer based in Cleveland, Ohio.

Read more about me, my work or my development practices.

Share this post