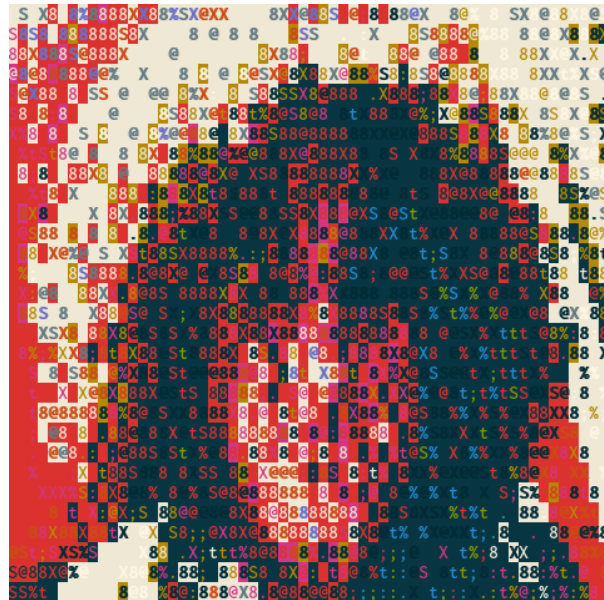


What the tmux?

Dotfiles Indy Meetup | 08.2020 | Clayton Parker

# Who Am I?



@claytron on the internets and IRL

My dotfiles

# What is it?

tmux == Terminal MUltipleXer

Many terminals within a terminal

A way to save your work for later or organize things locally

A better **screen**

# tmux VS screen

## Pros

- Client / server model
- Better keybinding support
- Multiple paste buffers
- More modern overall
- BSD licensed

## Cons

- No serial / telnet terminal
- No support for older platforms and odd terminals

# How it works?

Let's take a tour of the basic functionality of tmux

# Key Bindings

A note on the `prefix` before we get started

The `prefix` in tmux is `ctrl + b`

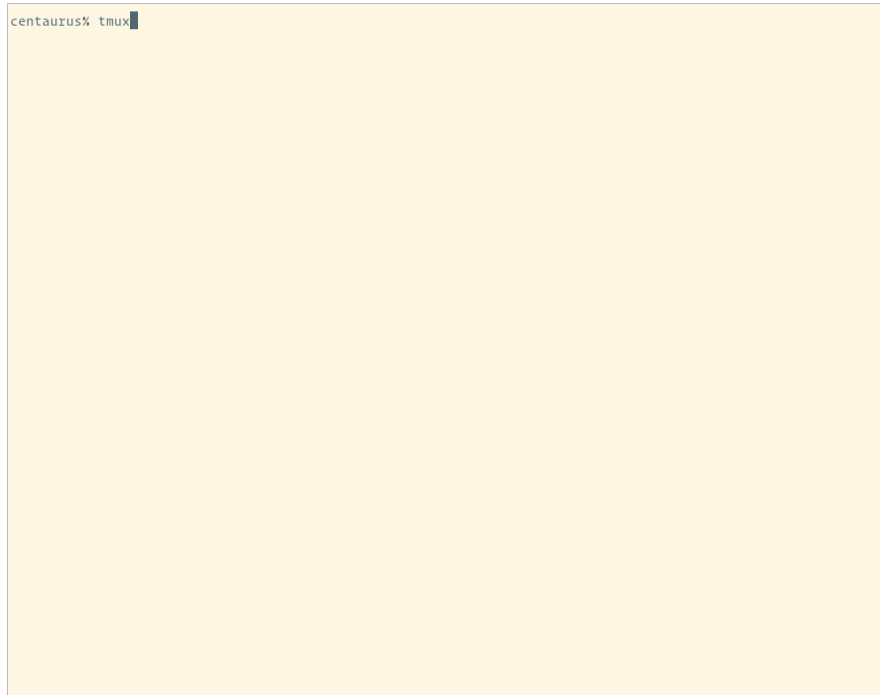
To avoid a conflict with screen's `ctrl + a`

Remap it in your `tmux.conf` if needed

```
# Change the prefix and release ctrl + b
set -g prefix C-a
unbind C-b

# Ability to send ctrl + a to applications still
bind C-a send-prefix
```

# Starting a new session



The foundation of **tmux** is the session

Every time you invoke **tmux** you are starting a new session within that server

# Sessions



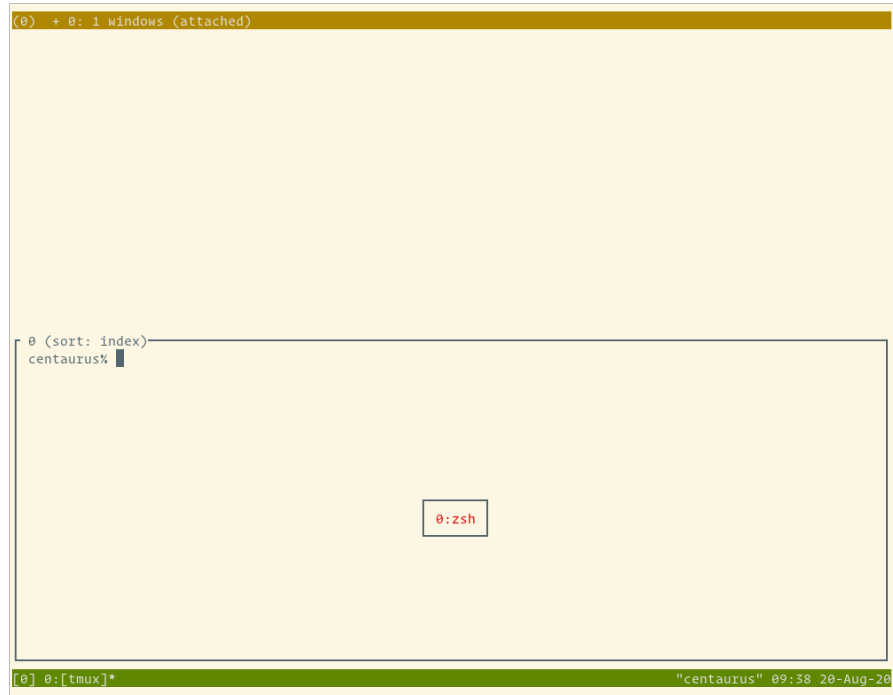
A session is started with a single window

A status line appears giving you info about the session

Another difference from default screen, automatic status line



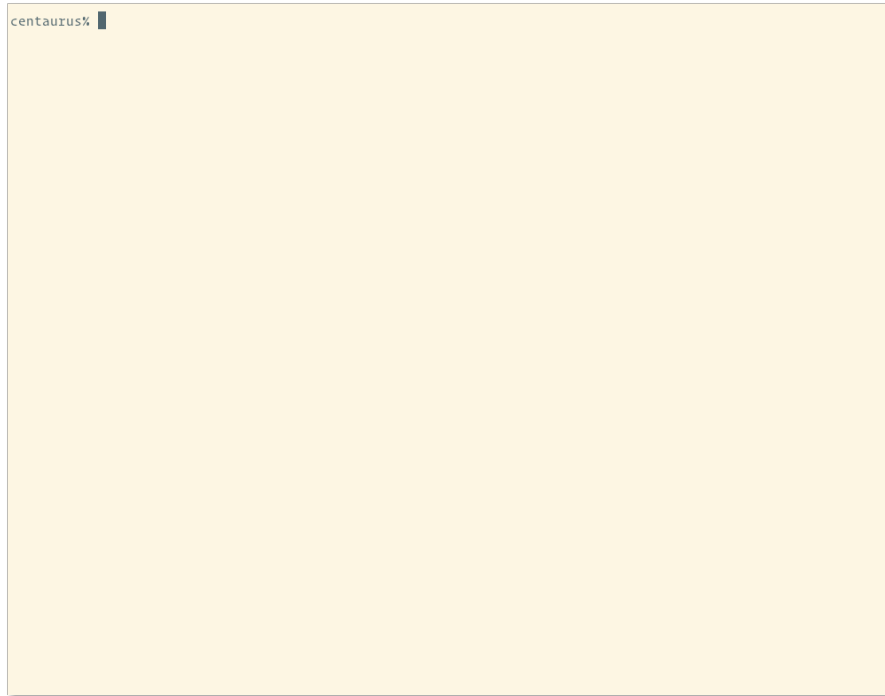
# Sessions



prefix + s

Now we can see we have one session running one window

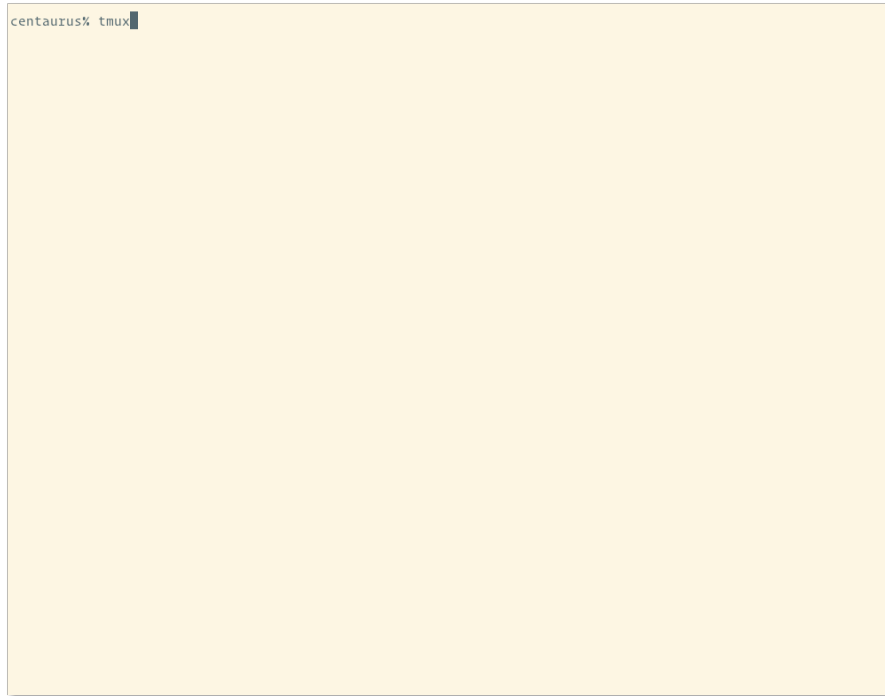
# Detaching



prefix + d

Now we are back to our original terminal, no longer in tmux

# Starting another session



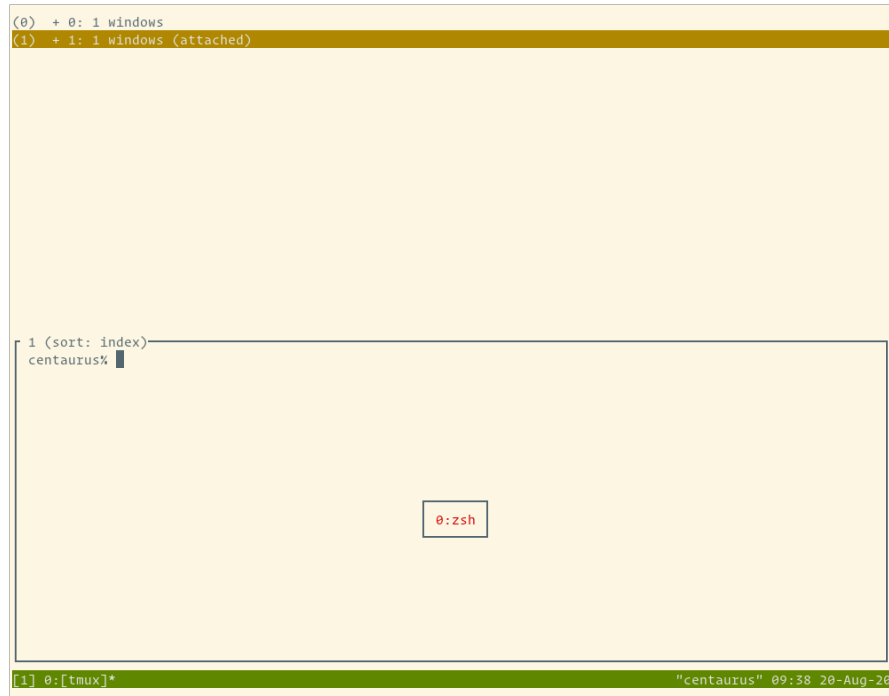
Running **tmux** again starts a completely new session on that server

# Sessions



We can see the session number is incremented in the status line

# Sessions

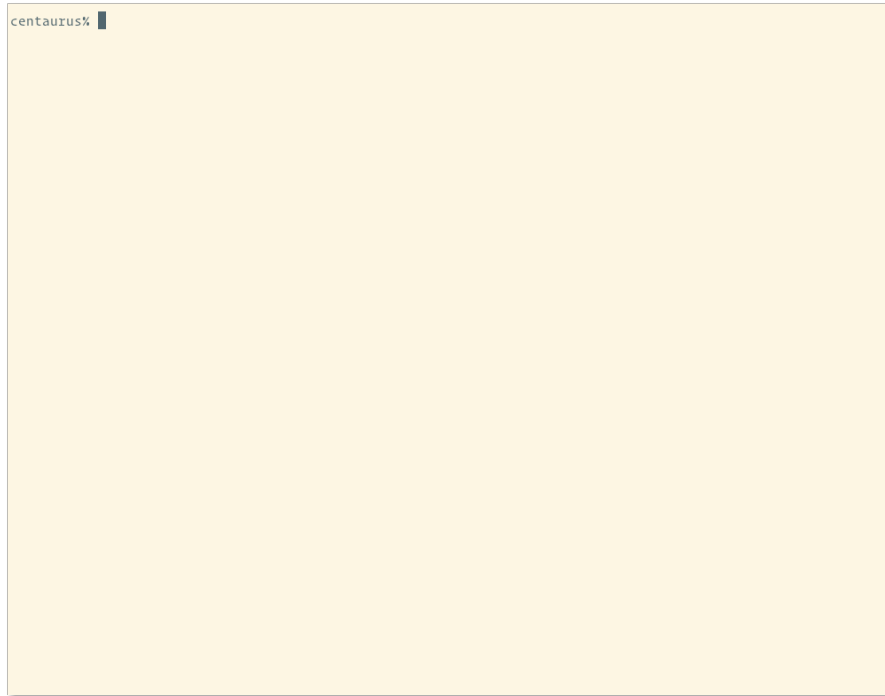


prefix + s

Now we can see we have two sessions running

This view also allows keyboard navigation, so you can see more detailed info

# Attaching



Now let's go back to our non tmux terminal

How do we get back to the original session?

# Attaching

```
centaurus% tmux list-sessions
0: 1 windows (created Thu Aug 20 09:37:20 2020)
1: 2 windows (created Thu Aug 20 09:38:31 2020) (attached)
centaurus%
centaurus% tmux attach-session -t 0
```

You can see a list of sessions with the **list-sessions** command (or **ls** for short)

You can attach to a specific session using the **attach** command

The **-t** is the **target**

# Attaching



Now we are back to the original session



# Windows and Panes

The next most fundamental thing about tmux

Windows are a collection of panes

Each pane is a horizontal or vertical split of the window

# Windows and Panes



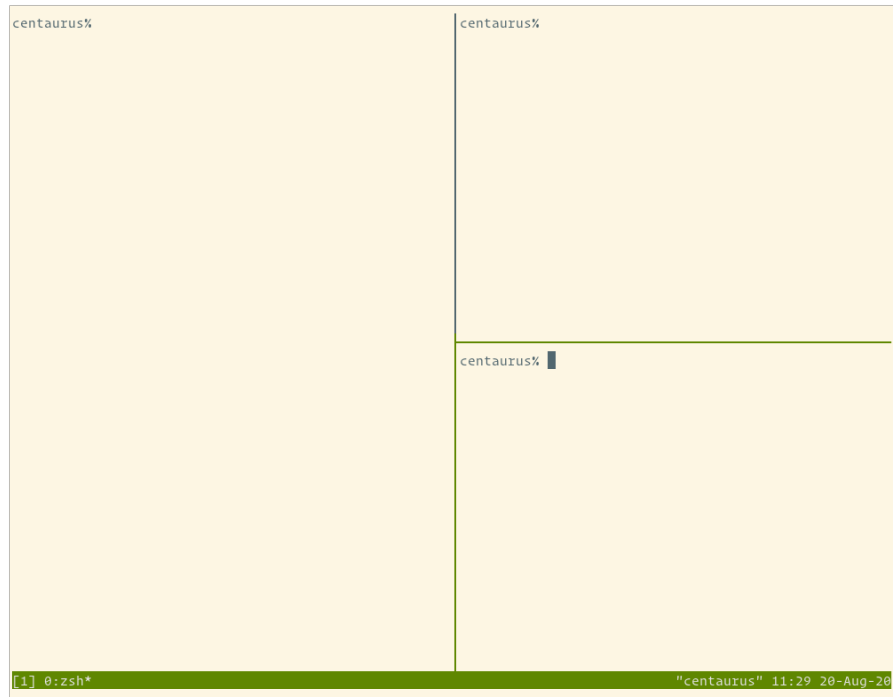
Starting out with one window in a new session

# Split Horizontally



prefix + %

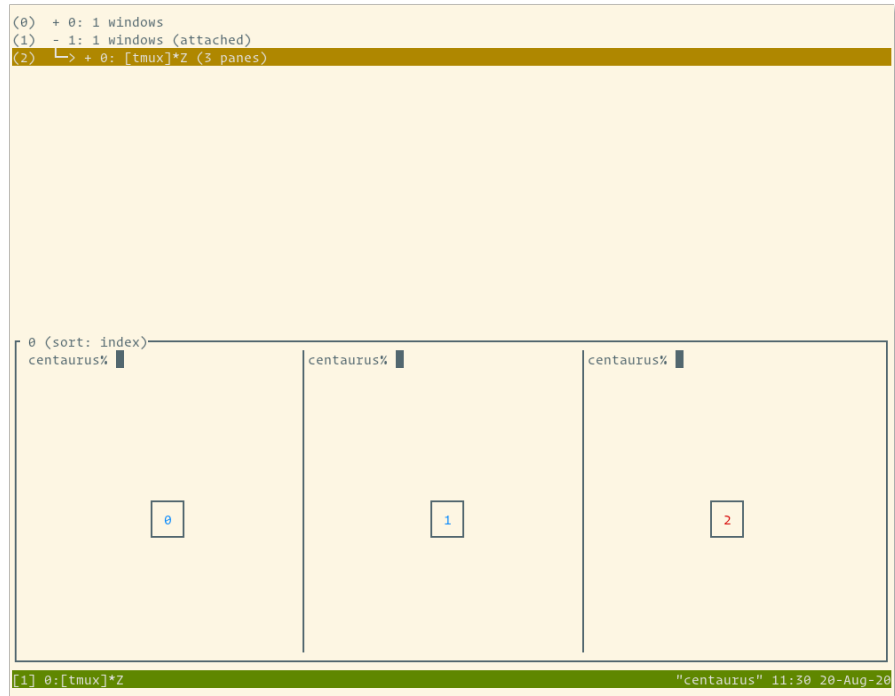
# Split Vertically



prefix + "

# Panes

```
(0) + 0: 1 windows
(1) - 1: 1 windows (attached)
(2)  ↳ + 0: [tmux]*Z (3 panes)
```



```
0 (sort: index)
centaurus% █ centaurus% █ centaurus% █
```

0 1 2

[1] 0:[tmux]\*Z "centaurus" 11:30 20-Aug-20

prefix + s

Now we can see all the panes via the session list

# New Window

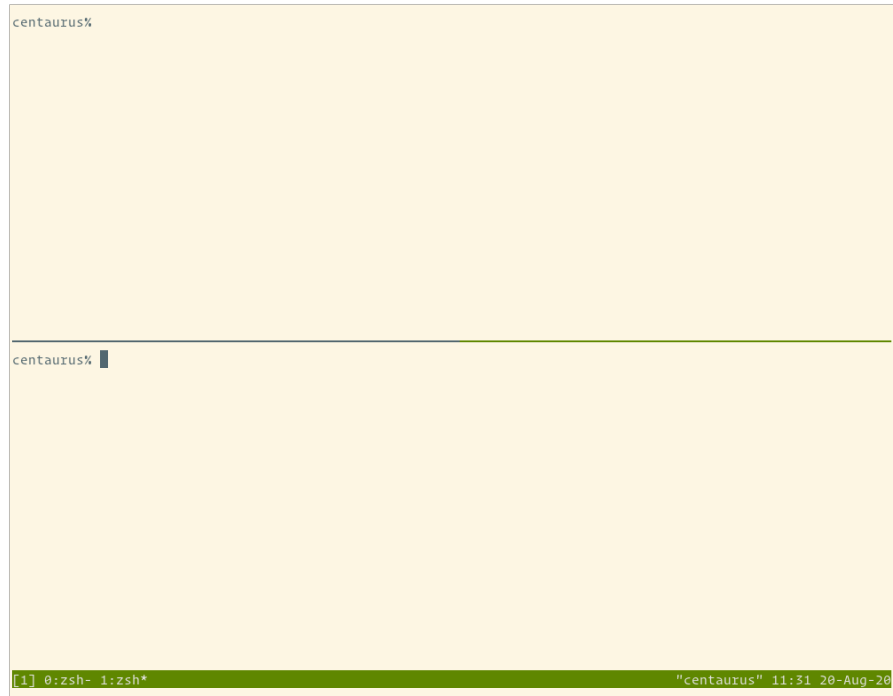


prefix + c

Create a second window in this session

You can see the new window in the status line

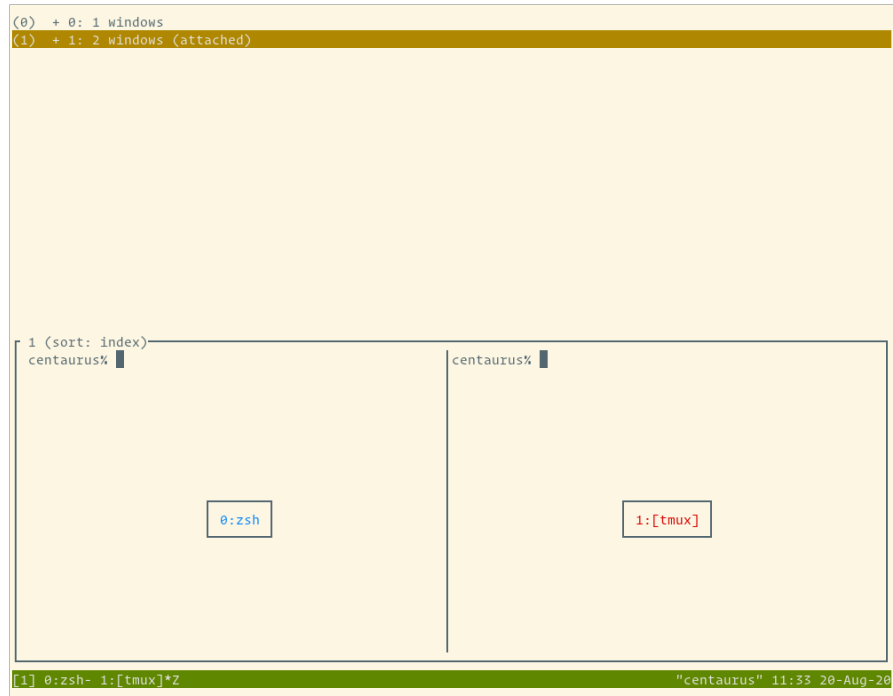
# New Window



prefix + "

And another split just for fun

# Session List



The screenshot shows a terminal window with a yellow background. At the top, there are two status lines: `(0) + 0: 1 windows` and `(1) + 1: 2 windows (attached)`. Below these, the main area is split into two panes. The left pane shows a prompt `centaurus%` and a small box labeled `0:zsh`. The right pane shows a prompt `centaurus%` and a small box labeled `1:[tmux]`. At the bottom, there is a green status bar with the text `[1] 0:zsh- 1:[tmux]*Z` on the left and `"centaurus" 11:33 20-Aug-20` on the right.

prefix + s

Go back to our session list and see everything we've created so far



```
(0) + 0: 1 windows
(1) - 1: 2 windows (attached)
(2)   |> + 0: zsh- (3 panes)
(3)   |> + 1: [tmux]*Z (2 panes)
```

```
1 (sort: index)
centaurus% █
```

0:zsh

1:[tmux]

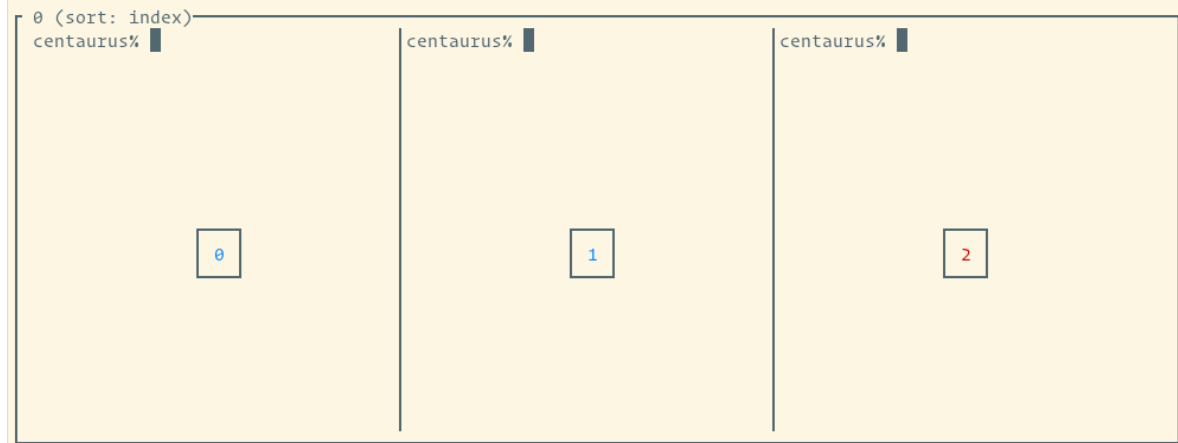
  

```
[1] 0:zsh- 1:[tmux]*Z "centaurus" 11:33 20-Aug-20
```

Using the keyboard to navigate

Hitting left and right to open / close the tree

```
(0) + 0: 1 windows  
(1) - 1: 2 windows (attached)  
(2) └─> + 0: zsh- (3 panes)  
(3) └─> + 1: [tmux]*Z (2 panes)
```



[1] 0:zsh- 1:[tmux]\*Z

"centaurus" 11:33 20-Aug-20

```
(0) + 0: 1 windows
(1) - 1: 2 windows (attached)
(2) -> - 0: zsh- (3 panes)
(3)   |> 0: zsh "centaurus"
(4)   |> 1: zsh "centaurus"
(5)   |> 2: zsh "centaurus"
(6) -> + 1: [tmux]*Z (2 panes)
```

```
0 (sort: index)
centaurus% █ centaurus% █ centaurus% █
```

0	1	2
---	---	---

```
[1] 0:zsh- 1:[tmux]*Z "centaurus" 11:33 20-Aug-20
```

```
(0) + 0: 1 windows
(1) - 1: 2 windows (attached)
(2) └─ - 0: zsh- (3 panes)
(3)   └─ 0: zsh "centaurus"
(4)   └─ 1: zsh "centaurus"
(5)   └─ 2: zsh "centaurus"
(6) └─ + 1: [tmux]*Z (2 panes)
```

1 (sort: index)─

centaurus% █

0

centaurus% █

1

[1] 0:zsh- 1:[tmux]\*Z

"centaurus" 11:33 20-Aug-20

```
(0) + 0: 1 windows
(1) - 1: 2 windows (attached)
(2) └─ - 0: zsh- (3 panes)
(3)   └─> 0: zsh "centaurus"
(4)   └─> 1: zsh "centaurus"
(5)   └─> 2: zsh "centaurus"
(6) └─ - 1: [tmux]*Z (2 panes)
(7)   └─> 0: zsh "centaurus"
(8)   └─> 1: zsh "centaurus"
```

```
1 (sort: index)─
centaurus% █
```

0

```
centaurus% █
```

1

```
[1] 0:zsh- 1:[tmux]*Z
```

```
"centaurus" 11:33 20-Aug-20
```

```
(0) + 0: 1 windows
(1) - 1: 2 windows (attached)
(2) └─ - 0: zsh- (3 panes)
(3)   │   └─ 0: zsh "centaurus"
(4)   │   └─ 1: zsh "centaurus"
(5)   │   └─ 2: zsh "centaurus"
(6) └─ - 1: [tmux]*Z (2 panes)
(7)   │   └─ 0: zsh "centaurus"
(8)   │   └─ 1: zsh "centaurus"
```

```
0 (sort: index)─
centaurus% █
```

0:zsh

```
[1] 0:zsh- 1:[tmux]*Z
```

```
"centaurus" 11:33 20-Aug-20
```

```
(0) - 0: 1 windows
(1)   └─> 0: zsh* (1 panes) "centaurus"
(2) - 1: 2 windows (attached)
(3)   └─> - 0: zsh- (3 panes)
(4)       └─> 0: zsh "centaurus"
(5)       └─> 1: zsh "centaurus"
(6)       └─> 2: zsh "centaurus"
(7)   └─> - 1: [tmux]*Z (2 panes)
(8)       └─> 0: zsh "centaurus"
(9)       └─> 1: zsh "centaurus"
```

```
0 (sort: index)─
centaurus% █
```

0:zsh

```
[1] 0:zsh- 1:[tmux]*Z "centaurus" 11:33 20-Aug-20
```

Now you can use the index on the side to go to a specific pane

Let's select 5

# Pane Selected



5

Now back to the second session's first window, first split

Session 1 window 0 pane 1, all zero based of course



# Naming

Sessions and Windows can have names

Our previous examples are all using numbered indexes

Let's see how naming things can make that clearer

# Naming

```
# My tmux config uses folds in vim.
#
# zR    open all folds
# zM    close all folds
# za    toggle fold at cursor position
# zj    move down to start of next fold
# zk    move up to end of previous fold

# General tmux settings {{{1
# -----

# set the command prefix to match screen
set -g prefix C-a
unbind C-b
# Allow for C-a C-a to send it to the application
bind C-a send-prefix

# Set the proper terminal type
set -g default-terminal "tmux"

# Set the delay so that it doesn't interfere with applications like
# vim
set -sg escape-time 0

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1

# use vi key bindings
setw -g mode-keys vi

# turn on mouse mode
setw -g mouse on
# Don't mark the window on right click
unbind -T root MouseDown3Pane
@

[0] 0:vim* "centaurus" 14:18 20-Aug-20
```

Let's say we have this in our first session

# Naming

```
print('hello')
```

```
centaurus% tail -f some.log
```

```
centaurus% python3 hello.py
hello
centaurus%
```

```
[1] 0:zsh* 1:python3- "centaurus" 14:18 20-Aug-20
```

And this in the second

# Naming

```
print('hello')
```

```
centaurus% tail -f some.log
```

```
centaurus% python3 hello.py
hello
centaurus%
```

```
(rename-session) work
```

prefix + \$

Let's name this one work

# Naming

```
print('hello')
```

```
centaurus% tail -f some.log
```

```
centaurus% python3 hello.py
hello
centaurus%
```

```
[work] 0:zsh* 1:python3- "centaurus" 14:19 20-Aug-20
```

Now it is starting to make more sense

# Naming

```
print('hello')
```

```
centaurus% tail -f some.log
```

```
centaurus% python3 hello.py
hello
centaurus%
```

```
(rename-window) hello world
```

prefix + ,

We can also rename the windows

This will stop the current process from renaming them constantly

# Naming

```
print('hello')
```

```
centaurus% tail -f some.log
```

```
centaurus% python3 hello.py
hello
centaurus%
```

```
[work] 0:hello world* 1:python3- "centaurus" 14:20 20-Aug-20
```

## Starting to look better

# Naming

```
~
~
~
~

>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
lines 1-11/11 (END)

centaurus%
```

(rename-window) docs

prefix + ,

Rinse and repeat



# Naming

```
~
~
~
~

>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
lines 1-11/11 (END)

centaurus%

[work] 0:hello world- 1:docs* "centaurus" 14:20 20-Aug-20
```

Looking tidy now

# Naming

```
# My tmux config uses folds in vim.
#
# zR    open all folds
# zM    close all folds
# za    toggle fold at cursor position
# zj    move down to start of next fold
# zk    move up to end of previous fold

# General tmux settings {{{1
# -----

# set the command prefix to match screen
set -g prefix C-a
unbind C-b
# Allow for C-a C-a to send it to the application
bind C-a send-prefix

# Set the proper terminal type
set -g default-terminal "tmux"

# Set the delay so that it doesn't interfere with applications like
# vim
set -sg escape-time 0

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1

# use vi key bindings
setw -g mode-keys vi

# turn on mouse mode
setw -g mouse on
# Don't mark the window on right click
unbind -T root MouseDown3Pane
@

[0] 0:vim* "centaurus" 14:18 20-Aug-20
```

Let's go back to the first session we created

# Naming

```
# My tmux config uses folds in vim.
#
# zR    open all folds
# zM    close all folds
# za    toggle fold at cursor position
# zj    move down to start of next fold
# zk    move up to end of previous fold

# General tmux settings {{{1
# -----

# set the command prefix to match screen
set -g prefix C-a
unbind C-b
# Allow for C-a C-a to send it to the application
bind C-a send-prefix

# Set the proper terminal type
set -g default-terminal "tmux"

# Set the delay so that it doesn't interfere with applications like
# vim
set -sg escape-time 0

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1

# use vi key bindings
setw -g mode-keys vi

# turn on mouse mode
setw -g mouse on
# Don't mark the window on right click
unbind -T root MouseDown3Pane
@

(rename-session) personal
```

prefix + \$

Same process as before for this window

# Naming

```
# My tmux config uses folds in vim.
#
# zR    open all folds
# zM    close all folds
# za    toggle fold at cursor position
# zj    move down to start of next fold
# zk    move up to end of previous fold

# General tmux settings {{{1
# -----

# set the command prefix to match screen
set -g prefix C-a
unbind C-b
# Allow for C-a C-a to send it to the application
bind C-a send-prefix
█
# Set the proper terminal type
set -g default-terminal "tmux"

# Set the delay so that it doesn't interfere with applications like
# vim
set -sg escape-time 0

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1

# use vi key bindings
setw -g mode-keys vi

# turn on mouse mode
setw -g mouse on
# Don't mark the window on right click
unbind -T root MouseDown3Pane
@

[personal]0:vim* "centaurus" 14:21 20-Aug-20
```

# Naming

```
# My tmux config uses folds in vim.
#
# zR    open all folds
# zM    close all folds
# za    toggle fold at cursor position
# zj    move down to start of next fold
# zk    move up to end of previous fold

# General tmux settings {{{1
# -----

# set the command prefix to match screen
set -g prefix C-a
unbind C-b
# Allow for C-a C-a to send it to the application
bind C-a send-prefix

# Set the proper terminal type
set -g default-terminal "tmux"

# Set the delay so that it doesn't interfere with applications like
# vim
set -sg escape-time 0

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1

# use vi key bindings
setw -g mode-keys vi

# turn on mouse mode
setw -g mouse on
# Don't mark the window on right click
unbind -T root MouseDown3Pane
@

(rename-window) dotfiles
```

prefix + ,

And again for the window

# Naming

```
# My tmux config uses folds in vim.
#
# zR    open all folds
# zM    close all folds
# za    toggle fold at cursor position
# zj    move down to start of next fold
# zk    move up to end of previous fold

# General tmux settings {{{1
# -----

# set the command prefix to match screen
set -g prefix C-a
unbind C-b
# Allow for C-a C-a to send it to the application
bind C-a send-prefix

# Set the proper terminal type
set -g default-terminal "tmux"

# Set the delay so that it doesn't interfere with applications like
# vim
set -sg escape-time 0

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1

# use vi key bindings
setw -g mode-keys vi

# turn on mouse mode
setw -g mouse on
# Don't mark the window on right click
unbind -T root MouseDown3Pane
@

[personal]@:dotfiles* "centaurus" 14:22 20-Aug-20
```

Now everything has a proper name

# Naming

```
(0) - personal: 1 windows (attached)
(1)   └─ 0: dotfiles* (1 panes) "centaurus"
(2) - work: 2 windows
(3)   └─ + 0: hello world- (3 panes)
(4)   └─ + 1: docs* (2 panes)
```

```
personal (sort: index)
# set the command prefix to match screen
set -g prefix C-a
unbind C-b
# Allow for C-a C-a to send it to the application
bind C-a send-prefix
█
# Set the proper terminal type
set -g default-terminal "tmux"

# Set the delay so that it doesn't interfere with applications like
# vim
set -sg escape-time 0

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1
```

[personal]0:dotfiles\* "centaurus" 14:22 20-Aug-20

prefix + s

Let's look at the list again for each session

# Naming

```
(0) ~ personal: 1 windows (attached)
(1)   ↳ 0: dotfiles* (1 panes) "centaurus"
(2) ~ work: 2 windows
(3)   ↳ + 0: hello world- (3 panes)
(4)   ↳ + 1: docs* (2 panes)
```

```
work (sort: index)
centaurus% python3 hello.py
hello
centaurus% █
```

0:hello world

1:docs

nes 1-11/11 (END)

[personal]0:dotfiles\* "centaurus" 14:22 20-Aug-20

Much more informative now



# Key Bindings

Vi and Emacs layouts for copy mode

Defaults are vague and hard to remember

Remap things to suit your style (read the Brian Hogan book)

# Pane Movement

By default movement between panes is

`prefix + <arrows>`

# Pane Movement

Set up vi style movement

```
bind C-h select-pane -L  
bind C-j select-pane -D  
bind C-k select-pane -U  
bind C-l select-pane -R
```

Allows to hold **ctrl** and use **hjkl**

# Pane Resizing

```
bind -r H resize-pane -L 5  
bind -r J resize-pane -D 5  
bind -r K resize-pane -U 5  
unbind L  
bind -r L resize-pane -R 5
```

Works much better for my vim brain

# Splitting

```
unbind '""'  
unbind %  
bind | split-window -h -c "#{pane_current_path}"  
bind - split-window -v -c "#{pane_current_path}"  
bind c new-window -c "#{pane_current_path}"
```

Another thing that works better for my brain

# Modality

Like vi, tmux has different modes

# Command Mode



prefix + :

Manipulate everything from here

Not often used, but handy for certain one off tasks

# Command Mode



A screenshot of a terminal window with a light yellow background. At the top left, there is a status line showing the number of windows and their creation times: `2: 1 windows (created Thu Aug 20 16:14:34 2020) (attached)`, `personal: 1 windows (created Thu Aug 20 12:12:29 2020) (attached)`, and `work: 2 windows (created Thu Aug 20 12:14:27 2020)`. In the top right corner, there is a small orange box containing the text `[0/0]`. The main area of the terminal is empty. At the bottom, there is a green status bar with the text `[2] 0:[tmux]*` on the left and `"centaurus" 16:46 20-Aug-20` on the right.

The output lands you in copy mode



# Copy Mode

```
# pane window client
# Put switch client back into place (was L)
unbind ""
bind -r "" switch-client -l

# Switch to last window
unbind ';'
bind -r ';' last-window

# Re-define switch to last pane, so this is easier to remember
bind -r l last-pane

# Appearance Settings {{{1
# -----

# Show notices when there is activity on another window
setw -g monitor-activity on
set -g visual-activity off

# Theme based on tmuxline
source-file ~/.tmux/tmuxline.tmux

# Status Line {{{1
# -----

# Force utf-8
set -gq status-utf8 on

# Localized settings for an OS {{{1
# -----

source-file ~/.tmux.local

# turn on folds
# vim: fdm=marker
centaurus%
[personal]0:dotfiles* "centaurus" 17:46 20-Aug-20
```

prefix + [

Probably my favorite feature, I still use tmux in a tiling window manager because of this

Modify the defaults to make it less aggravating

# Copy Mode

```
# pane window client [0/118]

# Put switch client back into place (was L)
unbind ""
bind -r "" switch-client -l

# Switch to last window
unbind ';'
bind -r ';' last-window

# Re-define swtich to last pane, so this is easier to remember
bind -r l last-pane

# Appearance Settings {{{1
# -----

# Show notices when there is activity on another window
setw -g monitor-activity on
set -g visual-activity off

# Theme based on tmuxline
source-file ~/.tmux/tmuxline.tmux

# Status Line {{{1
# -----

# Force utf-8
set -gq status-utf8 on

# Localized settings for an OS {{{1
# -----

source-file ~/.tmux.local

# turn on folds
# vim: fdm=marker
centaurus%
(search up) vi
```

?

Search the scrollback

# Copy Mode

```
# pane window client (1/22 results) [0/118]

# Put switch client back into place (was L)
unbind ""
bind -r "" switch-client -l

# Switch to last window
unbind ';'
bind -r ';' last-window

# Re-define switch to last pane, so this is easier to remember
bind -r l last-pane

# Appearance Settings {{{1
# -----

# Show notices when there is activity on another window
setw -g monitor-activity on
set -g visual-activity off

# Theme based on tmuxline
source-file ~/.tmux/tmuxline.tmux

# Status Line {{{1
# -----

# Force utf-8
set -gq status-utf8 on

# Localized settings for an OS {{{1
# -----

source-file ~/.tmux.local

# turn on folds
# 1m: fdm=marker
centaurus%
[personal]0:dotfiles* "centaurus" 17:24 20-Aug-20
```

Navigate the results with vi or emacs keybindings

# Copy Mode

```
set -sg escape-time 0 [89/118]

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1

# use vi key bindings
setw -g mode-keys vi

# turn on mouse mode
setw -g mouse on
# Don't mark the window on right click
unbind -T root MouseDown3Pane
bind -n WheelUpPane if-shell -F -t = "#{mouse_any_flag}" "send-keys -M" "if -Ft= '#{pane_in_mode}' 'send-keys -M' 'copy-mode -e'"
bind -n WheelDownPane select-pane -t= \; send-keys -M
# Default is to copy and cancel selection, this copies to the clipboard and leaves the selection
bind-key -T copy-mode-vi MouseDragEnd1Pane send-keys -X copy-pipe "pbcopy" \; display-message 'text copied to clipboard!'

# copy pasta
bind-key -T copy-mode-vi Enter send-keys -X copy-pipe "pbcopy" \; display-message 'Text copied to clipboard!'
bind-key -T copy-mode-vi y send-keys -X copy-pipe "pbcopy" \; display-message 'Text copied to clipboard!'
# vi like copy mode
# from https://twitter.com/nickmorrott/status/928343075646210050?s=09
bind -T copy-mode-vi v send-keys -X begin-selection
bind -T copy-mode-vi V send-keys -X select-line
bind -T copy-mode-vi C-v send-keys -X rectangle-toggle \; send-keys -X begin-selection
bind -T copy-mode-vi Escape send-keys -X cancel

# set the scrollbar
set -g history-limit 200000

# force tmux to use utf-8
setw -gq utf8 on

# Custom key bindings {{{1
[personal]0:dotfiles* "centaurus" 17:25 20-Aug-20
```

spacebar to start highlighting, enter to add it to the copy buffer

# Copy Mode

```
# pane window client
# Put switch client back into place (was L)
unbind ""
bind -r "" switch-client -l

# Switch to last window
unbind ';'
bind -r ';' last-window

# Re-define switch to last pane, so this is easier to remember
bind -r l last-pane

# Appearance Settings {{{1
# -----

# Show notices when there is activity on another window
setw -g monitor-activity on
set -g Visual-activity off

# Theme based on tmuxline
source-file ~/.tmux/tmuxline.tmux

# Status Line {{{1
# -----

# Force utf-8
set -gq status-utf8 on

# Localized settings for an OS {{{1
# -----

source-file ~/.tmux.local

# turn on folds
# vim: fdm=marker
centaurus%
:~list-buffers
```

Use command mode to show the copied text

# Copy Mode

```
buffer0: 90 bytes: "# Make window and pane indexes start with 1\012set -g base-index 1\012setw -g pane-ba[0/0]
dex 1\012"

[personal]0:dotfiles* "centaurus" 17:25 20-Aug-20
```

# Enhance Copy Mode

```
bind-key -T copy-mode-vi Enter send-keys -X copy-pipe "pbcopy" \; display-message  
bind-key -T copy-mode-vi y send-keys -X copy-pipe "pbcopy" \; display-message 'Tex  
  
# vi-ish bindings  
bind -T copy-mode-vi v send-keys -X begin-selection  
bind -T copy-mode-vi V send-keys -X select-line  
bind -T copy-mode-vi C-v send-keys -X rectangle-toggle \; send-keys -X begin-selec  
bind -T copy-mode-vi Escape send-keys -X cancel
```

Normally kicked out of copy mode, these help avoid that and retain position

Also, some vi style keys to make things easier

# Mouse Mode

```
# turn on mouse mode
setw -g mouse on

# Don't mark the window on right click
unbind -T root MouseDown3Pane
bind -n WheelUpPane if-shell -F -t = "#{mouse_any_flag}" "send-keys -M" "if -Ft= ';"
bind -n WheelDownPane select-pane -t= \; send-keys -M

# Default is to copy and cancel selection, this copys to the clipboard and leaves
bind-key -T copy-mode-vi MouseDragEnd1Pane send-keys -X copy-pipe "pbcopy" \; disp
```

The mouse is NOT evil!

Modify it to make it MUCH more useful



# More Config Examples

```
# Set the scrollbar
set -g history-limit 200000

# Set the delay so that it doesn't interfere with applications like vim
set -sg escape-time 0

# Make window and pane indexes start with 1
set -g base-index 1
setw -g pane-base-index 1

# Create a new session
bind S new-session
```

# Community

The community has created a lot of things that make tmux even better

See the awesome tmux list for more info on these things:

Plugins

Themes

Status line helpers

# Status line

```
1 # My tmux config uses folds in vim.
2 #
3 # zR    open all folds
4 # zM    close all folds
5 # zq    toggle fold at cursor position
6 # zi    move down to start of next fold
7 # zk    move up to end of previous fold
8 #
9 # General tmux settings {{{1
10 # -----
11 #
12 # set the command prefix to match screen
13 set -g prefix C-a
14 unbind C-b
15 # Allow for C-a C-a to send it to the application
16 bind C-a send-prefix
17
18 NORMAL > SPELL [EN] > +0 ~0 ~0 .tmux.conf tmux utf-8[unix] 1/145 |1
19
20 -----
21 ( My name is Zach Galifianakis, I hope I'm pronouncing that right. )
22 (
23   - Zach Galifianakis
24 )
25 -----
26
27      ^__^
28      o  (xx)\_____.
29         (__)\       )\/\
30         u   ||---w   ||
31             ||     ||
32
33 [ 16:34 Thu, Aug 20 — /home/clayton ]
34 [ centaurus% — ]
35
36 personal > i* > dotfiles 2020-08-20 < 16:34 centaurus
```

tmuxline making my status line easy to manage via vim and airline

# Managing Environments

Teamocil, Tmuxinator, [And many more...](#)

Used Teamocil for years (because of the Arrested Development joke)

Very helpful for repeated set ups locally or remote

# Clients

Language libraries like python, ruby, etc.

Interact with tmux via your own program

Endless possibilities for integration into things

# Pair Programming

Using tools like wemux to remote pair

Maybe not quite as easy as the VSCode pairing

Could be good for two command line junkies though

# Alternatives

Tiling Window Managers

Builtin to terminal (e.g. iTerm2, Kitty, etc)

# Links

- [My Dotfiles](#)
- [Awesome tmux](#)
- [wemux](#)
- [Teamocil](#)
- [Tmuxinator](#)

# Books

- Brian Hogan's [tmux 2 Productive Mouse-Free Development](#)
- [Tao of tmux](#) free online book

These books are both based on tmux 2, not the latest, but still very informative



**FIN**