From Screen to Supreme: The Evolution of Terminal Multiplexing

March 31, 2026 / Gavin Jackson

tmux terminal multiplexer zellij linux macos productivity cli iterm2

Once upon a time, we used `screen`. It worked. It kept processes running when you disconnected. But using it felt like operating heavy machinery without a manual.

Then came **tmux** — the terminal multiplexer that made terminal multiplexing actually pleasant. What started as a cleaner alternative to screen has become an essential tool for anyone who lives in the terminal.

This is part tutorial, part cheat sheet, and part love letter to a tool that changed how I work.

What Is a Terminal Multiplexer?

A terminal multiplexer lets you run multiple terminal sessions inside one window. Think of it as a window manager for your terminal.

What you can do:

- Split one terminal into multiple panes
- Create windows (tabs) and switch between them
- Detach from a session and leave everything running
- Reattach later from anywhere — even a different machine
- Share sessions with other users

The killer feature: Your work survives disconnection. Close your laptop, lose WiFi, battery dies — your processes keep running. Reattach and everything is exactly as you left it.

Installation

```
# macOS
brew install tmux

# Ubuntu/Debian
sudo apt install tmux
```

The Basics: Getting Started

Tmux uses a **prefix key** to send commands. By default, it's `Ctrl+b`. Press this, then the command key.

Sessions

A session is a collection of windows. Start one:

```
tmux # New session
tmux new -s myproject # Named session
tmux ls # List sessions
tmux attach # Attach to last session
tmux attach -t myproject # Attach to specific session
```

Inside tmux, prefix these commands:

Key	Action
`d`	Detach from session
`$`	Rename session
`s`	Session chooser (interactive list)
`(`	Previous session
`)`	Next session

Windows (Tabs)

Windows are like tabs in a browser. Each window fills the entire screen.

Key	Action
`c`	Create new window
`,`	Rename current window
`n`	Next window
`p`	Previous window
`0-9`	Switch to window number
`w`	Window chooser (interactive list)
`&`	Kill current window

Panes (Splits)

Panes let you see multiple terminals at once.

Key	Action
`%`	Split vertically (left/right)
`"`	Split horizontally (top/bottom)
`o`	Cycle through panes
`;`	Toggle to last active pane
`x`	Kill current pane
`z`	Zoom pane (fullscreen toggle)
`q`	Show pane numbers (press number to jump)
`{`	Swap with previous pane
`}`	Swap with next pane

Directional pane movement:

Key	Action
`↑`	Move up
`↓`	Move down
`←`	Move left
`→`	Move right

Hold `Ctrl+b`, then press arrow keys.

Resizing Panes

Hold `Ctrl+b`, then:

Key	Action
`Ctrl+↑`	Resize up
`Ctrl+↓`	Resize down
`Ctrl+←`	Resize left
`Ctrl+→`	Resize right

Or use the mouse if enabled (see configuration below).

Copy Mode

Tmux has its own scrollback buffer. Enter copy mode to scroll, search, and copy text.

Key	Action
`[`	Enter copy mode
`]`	Paste from buffer
`=`	Choose buffer to paste

In copy mode:

Key	Action
`↑/↓` or `PgUp/PgDn`	Scroll
`/`	Search down
`?`	Search up
`n`	Next search result
`N`	Previous search result
`Space`	Start selection
`Enter`	Copy selection
`q`	Quit copy mode

Configuration

Create `~/.tmux.conf` to customize tmux.

Sensible Defaults

```
# Change prefix to Ctrl+a (easier to reach, screen-compatible)
unbind C-b
set -g prefix C-a
bind C-a send-prefix

# Enable mouse support
set -g mouse on

# Start window numbering at 1
set -g base-index 1
setw -g pane-base-index 1

# Renumber windows when one is closed
set -g renumber-windows on

# Increase scrollback buffer
set -g history-limit 10000

# Use 256 colors
set -g default-terminal "screen-256color"

# Status bar styling
set -g status-style bg=black,fg=white
set -g window-status-current-style bg=blue,fg=white,bold

# Easier pane splitting
bind | split-window -h -c "#{pane_current_path}"
bind - split-window -v -c "#{pane_current_path}"
unbind '"'
unbind %

# Reload config with prefix + r
bind r source-file ~/.tmux.conf \; display "Config reloaded!"

# Vim-style pane navigation
bind h select-pane -L
bind j select-pane -D
bind k select-pane -U
bind l select-pane -R

# Vim-style window navigation
bind -r C-h select-window -t :-
bind -r C-l select-window -t :+

# Faster key repetition
set -s escape-time 0

# Activity monitoring
setw -g monitor-activity on
set -g visual-activity on
```

Modern Tmux (Tmux Plugin Manager)

Install TPM for easy plugin management:

```
git clone https://github.com/tmux-plugins/tpm ~/.tmux/plugins/tpm
```

Add to `~/.tmux.conf`:

```
# List of plugins
set -g @plugin 'tmux-plugins/tpm'
set -g @plugin 'tmux-plugins/tmux-sensible'
set -g @plugin 'tmux-plugins/tmux-resurrect'
set -g @plugin 'tmux-plugins/tmux-continuum'
set -g @plugin 'tmux-plugins/tmux-yank'

# Resurrect saves and restores sessions
set -g @resurrect-capture-pane-contents 'on'

# Continuum auto-saves every 15 minutes
set -g @continuum-restore 'on'

# Initialize TPM (keep at bottom)
run '~/.tmux/plugins/tpm/tpm'
```

Install plugins: `Ctrl+b` then `I` (capital I)

Key plugins:

- **tmux-sensible** — Sensible defaults everyone agrees on
- **tmux-resurrect** — Save and restore sessions after reboot
- **tmux-continuum** — Auto-save sessions periodically
- **tmux-yank** — Copy to system clipboard

---

iTerm2 Integration

iTerm2 on macOS has native tmux integration that's genuinely impressive. Instead of tmux drawing its own interface with ASCII borders, iTerm2 renders tmux windows as native tabs and panes.

How It Works

When you run tmux with the `-CC` (control center) flag, iTerm2 takes over:

- Tmux windows become iTerm2 **tabs**
- Tmux panes become iTerm2 **split panes**
- You use iTerm2's native keyboard shortcuts
- The visual result is seamless — no more ASCII borders

Starting Integrated Mode

```
tmux -CC new -s mysession # New session
tmux -CC attach -t mysession # Attach to existing
```

Or configure iTerm2 to always use integration:

```
iTerm2 → Preferences → General → tmux
 Open tmux windows as native tabs in a new window
 Automatically bury the tmux client session after connecting
```

What You Get

- **Native tabs** — Cmd+T creates new tmux windows
- **Native splits** — Cmd+D (vertical), Cmd+Shift+D (horizontal)
- **Mouse integration** — Drag to resize panes, click to focus
- **Better rendering** — No ASCII art borders, proper anti-aliasing
- **Reconnect support** — Disconnect and reconnect keeps your layout

The Trade-off

iTerm2 integration only works when running tmux locally or through SSH from iTerm2. If you SSH into a server and run tmux there without the `-CC` flag, you get standard tmux. This is usually fine — the integration shines for local development or when you're primarily working on one remote host.

Quick Cheat Sheet

Sessions

```
tmux new -s name # New named session
tmux attach -t name # Attach to session
tmux attach -d -t name # Attach, detach others
tmux ls # List sessions
tmux kill-session -t name # Kill session
tmux rename-session -t old new
```

Inside Tmux (Prefix: Ctrl+b)

Key	Action
Session	
`d`	Detach
`$`	Rename session
`s`	Session chooser
Windows	
`c`	New window
`,`	Rename window
`n`	Next window
`p`	Previous window
`0-9`	Go to window
`w`	Window chooser
`&`	Kill window
Panes	
`%`	Split vertical
`"`	Split horizontal
`o`	Next pane
`;`	Last pane
`x`	Kill pane
`z`	Zoom pane
`q`	Show pane numbers
`Space`	Cycle layouts
Copy Mode	
`[`	Enter copy mode
`]`	Paste
`=`	Choose buffer

Command Mode

Press `Ctrl+b` then `:` to enter command mode.

```
:set mouse on # Enable mouse
:set mouse off # Disable mouse
:set -g status off # Hide status bar
:set -g status on # Show status bar
:resize-pane -D 10 # Resize down 10 cells
:swap-pane -U # Swap with previous pane
:join-pane -s :1 # Join window 1 to current
:break-pane # Break pane to new window
```

Advanced Tips

Synchronize Panes

Send the same input to all panes in a window — useful for running commands on multiple servers:

```
Ctrl+b :setw synchronize-panes on
```

Type once, appears everywhere. Turn off with `off`.

Pair Programming

Two users can attach to the same session:

```
# User 1
tmux -S /tmp/shared new -s pair
chmod 777 /tmp/shared

# User 2
tmux -S /tmp/shared attach
```

Both see and control the same terminal.

Tmuxinator (Session Management)

Define sessions in YAML, start them with one command:

```
gem install tmuxinator
mux new myproject
```

Edit `~/.config/tmuxinator/myproject.yml`:

```
name: myproject
root: ~/projects/myproject

windows:
 - editor:
 layout: main-vertical
 panes:
 - vim
 - git status
 - server:
 panes:
 - rails server
 - logs:
 panes:
 - tail -f log/development.log
```

Start it: `mux myproject`

Zellij: The New Contender

Zellij is a Rust-based terminal multiplexer that reimagines what a modern multiplexer should be. It's not just a tmux clone — it has genuinely innovative features.

What Makes Zellij Different

Mode-based navigation: Instead of a prefix key, Zellij uses modes. Press `Ctrl+g` to enter "locked" mode, `Ctrl+p` for pane mode, `Ctrl+t` for tab mode. Each mode has its own keybindings displayed on screen.

Built-in UI: Zellij shows a toolbar at the bottom with available commands. No more forgetting keybindings — they're always visible.

Layouts: Define complex pane arrangements in YAML and load them instantly:

```
# ~/.config/zellij/layouts/dev.kdl
layout {
 pane split_direction="vertical" {
 pane {
 command "nvim"
 }
 pane split_direction="horizontal" {
 pane
 pane
 }
 }
}
```

Floating panes: Pop a pane out as an overlay, like a modal dialog. Close it and you're back where you were.

Stacked panes: Stack panes in one slot, switch between them like tabs within a tab.

WebAssembly plugins: Write plugins in any language that compiles to WASM. There's already a growing ecosystem — file explorers, resource monitors, git dashboards.

Installation

```
# macOS
brew install zellij

# Linux
cargo install zellij
# Or download binary from GitHub releases
```

Quick Start

```
zellij # New session
zellij attach # Attach to existing
zellij --layout dev # Start with a layout
```

Default keybindings:

Mode	How to Enter	What You Can Do
Normal	Default	Navigate panes with arrows
Locked	`Ctrl+g`	Pass keys through to terminal
Pane	`Ctrl+p`	Split, resize, close panes
Tab	`Ctrl+t`	Create, switch, rename tabs
Resize	`Ctrl+n`	Resize panes with arrows
Move	`Ctrl+h`	Move panes between positions
Search	`Ctrl+s`	Search scrollback
Session	`Ctrl+o`	Detach, switch sessions

When to Choose Zellij Over Tmux

Choose Zellij if...	Stick with Tmux if...
You want discoverable keybindings	You have years of muscle memory
Layouts are important to your workflow	You need maximum compatibility
Floating/stacked panes appeal to you	You rely on extensive plugins
You want a modern Rust codebase	You need iTerm2 integration
WebAssembly plugins excite you	You need session resurrection now

Zellij's session resurrection is coming but not as mature as tmux-resurrect. The plugin system is promising but young.

Why Tmux Wins (For Now)

I've tried the alternatives:

- **Screen** — Functional but clunky
- **iTerm2 native tabs** — Great, but macOS only
- **Zellij** — Impressive, fast, innovative — watch this space

Tmux wins because it's everywhere, it's fast, and it does one thing exceptionally well: keep your terminal environment alive and organized.

The learning curve is real — those first few days of hitting `Ctrl+b` will feel awkward. But stick with it. Within a week, muscle memory takes over. Within a month, you'll wonder how you worked without it.

Resources

- Tmux GitHub
- Tmuxinator
- Tmux Plugin Manager
- iTerm2 Tmux Integration
- Tmux Cheat Sheet

Downloaded from https://www.gavinj.net/post/tmux-terminal-multiplexing
Generated October 10, 2026. Copyright Gavin Jackson. All rights reserved.