The Complete tmux Guide

Terminal Multiplexer Setup, Configuration & Usage

Robert Bergman

February 2026

---

tmux 3.6a · macOS / Linux

Generated February 26, 2026

Table of Contents

1 Introduction

1.1 What is tmux?

tmux (terminal multiplexer) lets you run multiple terminal sessions inside a single terminal window. It provides three key capabilities:

1. **Multiplexing** — Split your terminal into multiple panes and windows, each running independent processes.
2. **Persistence** — Sessions survive if your terminal closes or your SSH connection drops. Re-attach and pick up where you left off.
3. **Sharing** — Multiple users can attach to the same session for pair programming or demonstrations.

1.2 Key Concepts

Concept	Description
Server	Background process that manages all sessions. Starts automatically.
Session	A collection of windows. You can have many sessions running simultaneously.
Window	A single screen within a session. Like browser tabs.
Pane	A subdivision of a window. Each pane runs its own shell.
Prefix	A key combination pressed before tmux commands. Default: `Ctrl+b`. Our config uses: `Ctrl+a`
Client	Your terminal connection to a tmux session.

Table 1: tmux terminology

1.3 How It All Fits Together

```
tmux server
├── Session: "work"
│   ├── Window 1: "editor"    ← currently active
│   │   ├── Pane 1: vim       ← focused
│   │   ├── Pane 2: terminal
│   │   └── Pane 3: logs
│   ├── Window 2: "git"
│   │   └── Pane 1: shell
│   └── Window 3: "tests"
│       └── Pane 1: test runner
└── Session: "personal"
    └── Window 1: "scratch"
        └── Pane 1: shell
```

2 Installation

2.1 macOS (Homebrew)

```
brew install tmux
```

2.2 Linux

```
# Debian / Ubuntu
sudo apt-get update && sudo apt-get install -y tmux

# Fedora / RHEL
sudo dnf install -y tmux

# Arch Linux
sudo pacman -Sy tmux
```

2.3 Verify Installation

```
tmux -V
# Expected output: tmux 3.6a (or similar)
```

2.4 Using the Install Script

The included `scripts/install-tmux.sh` automates installation across platforms:

```
chmod +x scripts/install-tmux.sh
./scripts/install-tmux.sh
```

3 Configuration

tmux reads its configuration from `~/.tmux.conf` on startup. The included setup script generates a fully-documented config file.

3.1 Running the Setup Script

```
chmod +x scripts/setup-tmux-config.sh
./scripts/setup-tmux-config.sh
```

This creates `~/.tmux.conf` (backing up any existing config first).

3.2 Configuration Walkthrough

3.2.1 Prefix Key

The default prefix `Ctrl+b` is awkward to reach. Our config remaps it to `Ctrl+a`:

```
unbind C-b
set -g prefix C-a
bind C-a send-prefix
```

Tip: To send a literal `Ctrl+a` to a program inside tmux (e.g., to jump to beginning of line in bash), press `Ctrl+a` twice.

3.2.2 Terminal Colors

For proper 256-color and true-color support:

```
set -g default-terminal "tmux-256color"
set -ag terminal-overrides ",xterm-256color:RGB"
```

3.2.3 Mouse Support

Our config enables mouse for:

- Pane selection (click to focus)
- Pane resizing (drag borders)
- Window selection (click on status bar)
- Scrolling (scroll wheel enters copy mode)

```
set -g mouse on
```

3.2.4 Intuitive Split Keys

Instead of the cryptic defaults (`"` and `%`), our config uses:

Key	Action
`prefix + |`	Split pane horizontally (side by side)
`prefix + -`	Split pane vertically (top/bottom)

Table 2: Intuitive split bindings

3.2.5 Status Bar

The status bar is positioned at the top with a Catppuccin-inspired color scheme:

- **Left:** Session name (pink badge)
- **Center:** Window list with active highlight (blue)
- **Right:** Date and time

3.3 Configuration Reference

Every setting in the generated `.tmux.conf` is documented with inline comments. Key sections:

Section	What It Configures
General Settings	Prefix, colors, base index, history, mouse, escape time
Window Management	Config reload, splits, new window path
Pane Navigation	Vim-style and Alt+Arrow switching
Pane Resizing	Prefix + H/J/K/L resize by 5 cells
Window Navigation	Shift+Arrow switching, window swapping
Copy Mode	Vi-style selection, clipboard integration
Status Bar	Position, colors, format, update interval
Plugins	TPM plugin declarations (commented by default)

Table 3: Configuration sections

4 Essential Usage

4.1 Starting tmux

```
# Start a new unnamed session
tmux

# Start a new named session
tmux new-session -s work

# Start with a specific working directory
tmux new-session -s project -c ~/Projects/myapp
```

4.2 Sessions

Sessions are the top-level organizational unit. You typically create one session per project or context.

4.2.1 Creating and Managing Sessions

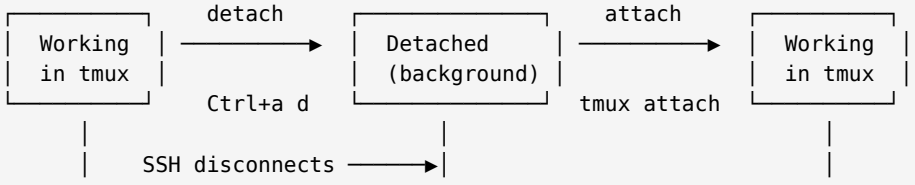
```
# From outside tmux:
tmux new -s <name>           # Create named session
tmux ls                      # List all sessions
tmux attach -t <name>        # Attach to a session
tmux kill-session -t <name>  # Destroy a session
tmux kill-server             # Kill ALL sessions

# From inside tmux (prefix commands):
```

Key	Action
`prefix + d`	Detach from current session
`prefix + s`	Interactive session list/switcher
`prefix + $`	Rename current session
`prefix + (`	Switch to previous session
`prefix + )`	Switch to next session

Table 4: Session key bindings

Tip: Detaching (`prefix + d`) leaves the session running in the background. All processes continue. Re-attach anytime with `tmux attach`.

4.2.2 The Session Workflow



```
|    Terminal closes ──────►|                          |
|    Laptop sleeps ──────►|                          |
```

4.3 Windows

Windows are like tabs within a session.

Key	Action
`prefix + c`	Create new window
`prefix + ,`	Rename current window
`prefix + &`	Close current window (with confirmation)
`prefix + n`	Next window
`prefix + p`	Previous window
`prefix + 0-9`	Go to window by number
`prefix + w`	Interactive window/session tree
`Shift + Left`	Previous window (no prefix needed)
`Shift + Right`	Next window (no prefix needed)
`prefix + <`	Swap window left
`prefix + >`	Swap window right

Table 5: Window key bindings

4.4 Panes

Panes split a window into multiple terminal areas.

4.4.1 Creating Panes

Key	Action
`prefix + |`	Split horizontally (left/right)
`prefix + -`	Split vertically (top/bottom)
`prefix + x`	Close current pane
`prefix + z`	Toggle zoom (fullscreen pane)
`prefix + !`	Break pane out into its own window
`prefix + q`	Flash pane numbers — press a number to jump

Table 6: Pane management bindings

4.4.2 Navigating Panes

Key	Action
`prefix + h`	Move left
`prefix + j`	Move down
`prefix + k`	Move up
`prefix + l`	Move right
`Alt + Arrow`	Move in arrow direction (no prefix)

Table 7: Pane navigation bindings

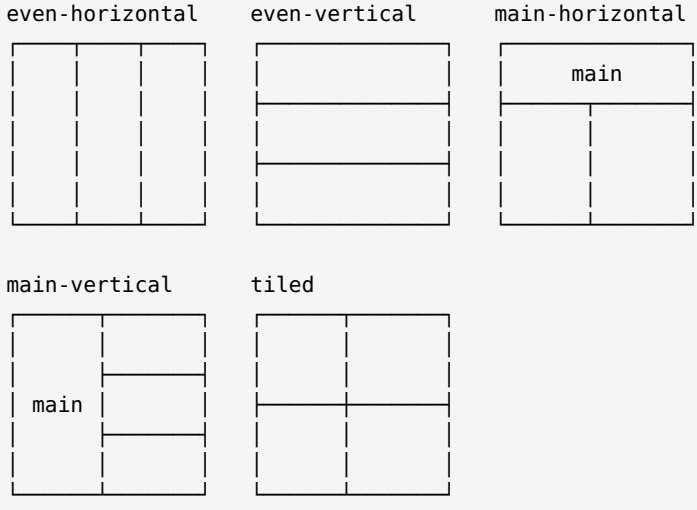
4.4.3 Resizing Panes

Key	Action
`prefix + H`	Grow left by 5 cells
`prefix + J`	Grow down by 5 cells
`prefix + K`	Grow up by 5 cells
`prefix + L`	Grow right by 5 cells
`prefix + Space`	Cycle through preset layouts

Table 8: Pane resizing bindings

4.4.4 Pane Layouts

tmux has five built-in layouts. Cycle through them with `prefix + Space`.



5 Copy Mode

Copy mode lets you scroll through terminal output and copy text using vi-style keybindings.

5.1 Entering Copy Mode

- `prefix + v` — Enter copy mode (custom binding)
- `prefix + [` — Enter copy mode (default binding)
- Scroll up with mouse wheel (if mouse is enabled)

5.2 Navigating in Copy Mode

Key	Action
`h j k l`	Move cursor (vi-style)
`w / b`	Forward / back by word
`Ctrl+u / Ctrl+d`	Half-page up / down
`Ctrl+b / Ctrl+f`	Full page up / down
`g / G`	Jump to top / bottom
`/ or ?`	Search forward / backward
`n / N`	Next / previous search match

Table 9: Copy mode navigation

5.3 Selecting and Copying

1. Press `v` to start selection
2. Move cursor to extend selection
3. Press `y` to copy (yanks to system clipboard on macOS)
4. Press `Ctrl+v` before `v` for rectangle/block selection

Tip: On macOS, our config pipes yanked text through `pbcopy`, so it goes straight to your system clipboard. On Linux, change `pbcopy` to `xclip -selection clipboard` or `wl-copy` (Wayland) in `.tmux.conf`.

5.4 Pasting

- `prefix + ]` — Paste the most recent buffer
- `tmux list-buffers` — View all copy buffers
- `tmux show-buffer` — Display the most recent buffer
- `tmux choose-buffer` — Interactive buffer picker

6 Command Mode

Press `prefix + :` to open the tmux command prompt at the bottom of the screen. This gives you access to every tmux command.

6.1 Useful Commands

```
# Window and pane manipulation
:new-window -n "logs"
:split-window -h -p 30         # horizontal split, 30% width
:swap-pane -D                  # swap pane downward
:join-pane -s 2 -t 1           # move pane from window 2 to window 1
:break-pane                    # move pane to its own window

# Session management
:new-session -s work
:rename-session production
:switch-client -t other

# Layout
:select-layout even-horizontal
:select-layout main-vertical
:resize-pane -R 10             # grow right 10 cells
:resize-pane -D 5              # grow down 5 cells

# Display
:set status off                # hide status bar
:set status on                 # show status bar
:display-message "#{pane_current_path}"

# Capture and save pane output
:capture-pane -S -3000         # capture last 3000 lines
:save-buffer ~/tmux-output.txt
```

7 Plugins with TPM

The Tmux Plugin Manager (TPM) makes it easy to install and manage plugins.

7.1 Installing TPM

```
chmod +x scripts/setup-tpm.sh
./scripts/setup-tpm.sh
```

This clones TPM and enables the plugin lines in your `.tmux.conf`.

7.2 Managing Plugins

Key	Action
`prefix + I`	Install plugins listed in `.tmux.conf`
`prefix + U`	Update all plugins
`prefix + Alt+u`	Remove unlisted plugins

Table 10: TPM key bindings

7.3 Recommended Plugins

7.3.1 tmux-resurrect

Saves and restores tmux sessions (windows, panes, working directories):

- `prefix + Ctrl+s` — Save session
- `prefix + Ctrl+r` — Restore session

7.3.2 tmux-continuum

Automatic session saving every 15 minutes. Works with tmux-resurrect:

```
set -g @continuum-restore 'on'    # auto-restore on tmux start
set -g @continuum-save-interval '15'
```

7.3.3 tmux-yank

Enhanced clipboard support across platforms. Automatically uses the right clipboard command for your OS.

7.4 Adding New Plugins

Add a plugin line to `~/.tmux.conf`:

```
set -g @plugin 'github-user/plugin-name'
```

Then press `prefix + I` to install.

8 Scripting & Automation

tmux's real power emerges when you script it.

8.1 Session Layouts

The included `scripts/tmux-layout-dev.sh` creates a development workspace:

```
./scripts/tmux-layout-dev.sh myproject ~/Projects/myapp
```

This creates:

```
Window 1 "editor":                Window 2 "git":     Window 3 "misc":
┌──────────────────┬────────────┐   ┌──────────────┐   ┌──────────────┐
│                  │ terminal   │   │              │   │              │
│   editor         │            │   │   git shell  │   │  free shell  │
│   (70%)          ├────────────┤   │              │   │              │
│                  │ watcher    │   │              │   │              │
│                  │ (40%)      │   │              │   │              │
└──────────────────┴────────────┘   └──────────────┘   └──────────────┘
```

8.2 The Sessionizer

The `scripts/tmux-sessionizer.sh` script provides fast project switching using `fzf`:

```
# Install fzf first
brew install fzf

# Run the sessionizer
./scripts/tmux-sessionizer.sh

# Or provide a directory directly
./scripts/tmux-sessionizer.sh ~/Projects/myapp
```

To bind it to a key in tmux, add to `.tmux.conf`:

```
bind f run-shell "path/to/tmux-sessionizer.sh"
```

> **Tip:** Copy `tmux-sessionizer.sh` to `~/.local/bin/` and edit the `SEARCH_DIRS` array to match your project directories.

8.3 Sending Commands to Panes

You can script tmux to send keystrokes to specific panes:

```
# Start a session and run commands in specific panes
tmux new-session -d -s dev -c ~/project
tmux send-keys -t dev:1 'vim .' Enter
tmux split-window -h -t dev:1 -c ~/project
tmux send-keys -t dev:1.2 'npm run dev' Enter
tmux split-window -v -t dev:1.2 -c ~/project
tmux send-keys -t dev:1.3 'npm test -- --watch' Enter
tmux attach -t dev
```

8.4 Target Syntax

tmux uses a target syntax to address sessions, windows, and panes:

```
session:window.pane

Examples:
  dev           → session "dev"
  dev:1         → session "dev", window 1
  dev:1.2       → session "dev", window 1, pane 2
  dev:editor    → session "dev", window named "editor"
  :1            → current session, window 1
  :.2           → current session, current window, pane 2
```

9 Advanced Techniques

9.1 Synchronized Panes

Send the same keystrokes to all panes in a window simultaneously:

```
# Toggle synchronized input
:setw synchronize-panes on
:setw synchronize-panes off
```

This is invaluable for running the same command on multiple servers.

9.2 Linking Windows

Share a window between sessions without duplicating it:

```
tmux link-window -s source_session:1 -t target_session:5
```

9.3 Hooks

tmux can run commands in response to events:

```
# Run a command whenever a new session is created
set-hook -g session-created 'display "Welcome!"'

# Run a command when a pane is closed
set-hook -g pane-died 'respawn-pane -k'

# Auto-rename windows based on running command
set-hook -g pane-focus-in 'rename-window "#{pane_current_command}"'
```

9.4 Environment Variables

tmux manages environment variables that are passed to new panes:

```
# Update environment on attach (useful for SSH agent forwarding)
set -g update-environment "SSH_AUTH_SOCK SSH_AGENT_PID DISPLAY"

# Set a variable in the tmux environment
tmux set-environment -g MY_VAR "value"

# Show tmux environment
tmux show-environment -g
```

9.5 Pipe Pane (Logging)

Capture all output from a pane to a file:

```
# Start logging current pane
:pipe-pane -o 'cat >> ~/tmux-pane-#{pane_id}.log'

# Stop logging
:pipe-pane
```

9.6 Format Strings

tmux has a powerful format string system for customizing output:

```
# List windows with custom format
tmux list-windows -F '#{window_index}: #{window_name} (#{window_panes} panes)'

# List panes with details
tmux list-panes -F '#{pane_index}: #{pane_current_command} [#{pane_width}x#{pane_height}]'

# Display information
tmux display -p '#{session_name}:#{window_index}.#{pane_index}'
```

10 Workflows

10.1 Remote Development over SSH

```
# On the remote server:
tmux new -s work

# ... do your work ...
# Connection drops? No problem.

# Reconnect and reattach:
ssh server
tmux attach -t work
```

Warning: If you nest tmux inside tmux (local + remote), press the prefix **twice** to send commands to the inner tmux. For example, `Ctrl+a Ctrl+a c` creates a window in the inner session.

10.2 Pair Programming

```
# Developer A creates a shared session
tmux new -s pair

# Developer B attaches (same machine / SSH)
tmux attach -t pair

# For independent cursors, use grouped sessions:
tmux new -s pair-b -t pair   # shares windows but independent view
```

10.3 IDE-Style Layout

Create a reproducible development environment:

```
#!/usr/bin/env bash
# ide.sh — IDE-like tmux layout
SESSION="ide"

tmux new-session -d -s $SESSION -n "code" -c ~/project
tmux send-keys -t $SESSION:code 'nvim .' Enter

tmux new-window -t $SESSION -n "term" -c ~/project
tmux split-window -v -t $SESSION:term -c ~/project -l '30%'
tmux send-keys -t $SESSION:term.1 'git status' Enter
tmux send-keys -t $SESSION:term.2 'npm run dev' Enter

tmux new-window -t $SESSION -n "db" -c ~/project
tmux send-keys -t $SESSION:db 'echo "Database console"' Enter

tmux select-window -t $SESSION:code
tmux attach -t $SESSION
```

11 Troubleshooting

11.1 Common Issues

11.1.1 Colors Look Wrong

```
# Verify your terminal supports 256 colors
echo $TERM
# Should show: xterm-256color or similar

# Inside tmux, verify:
tmux info | grep -i rgb
# Should show Tc or RGB capability
```

If colors are still wrong, ensure your terminal emulator (iTerm2, Alacritty, etc.) is set to report `xterm-256color`.

11.1.2 Escape Key Has a Delay

This is caused by tmux waiting to see if Escape is part of a multi-key sequence. Our config already fixes this:

```
set -sg escape-time 10
```

If you still experience delay, try setting it to `0`.

11.1.3 Copy/Paste Not Working

- **macOS:** Ensure `pbcopy` is available. Install `reattach-to-user-namespace` if on an older macOS.
- **Linux:** Install `xclip` or `xsel`, then update the copy command in `.tmux.conf`.

11.1.4 Session Not Found on Reattach

```
# List running sessions
tmux ls

# If the server died, sessions are lost (use tmux-resurrect to prevent this)
# Check if the server is running
pgrep -f "tmux: server"
```

11.1.5 "Terminal is not fully functional" Warnings

```
# Set TERM correctly before starting tmux
export TERM=xterm-256color
```

11.2 Useful Debug Commands

```
# Show all tmux options and their values
tmux show-options -g         # global options
tmux show-options -w         # window options
tmux show-options -s         # server options

# Show all key bindings
tmux list-keys

# Show tmux info (terminal capabilities)
tmux info

# Show all tmux commands
tmux list-commands
```

12 Quick Reference Card

All bindings below assume the custom config with `Ctrl+a` as prefix.

12.1 Sessions

Key / Command	Action
`tmux new -s name`	New named session
`tmux ls`	List sessions
`tmux attach -t name`	Attach to session
`tmux kill-session -t name`	Kill session
`prefix + d`	Detach
`prefix + s`	Session switcher
`prefix + $`	Rename session

12.2 Windows

Key	Action
`prefix + c`	New window
`prefix + ,`	Rename window
`prefix + &`	Close window
`prefix + n / p`	Next / previous
`prefix + 0-9`	Go to window N
`prefix + w`	Window tree
`Shift+Left / Right`	Prev / next (no prefix)

12.3 Panes

Key	Action
`prefix + |`	Split horizontal
`prefix + -`	Split vertical
`prefix + x`	Close pane
`prefix + z`	Toggle zoom
`prefix + h/j/k/l`	Navigate (vim)
`prefix + H/J/K/L`	Resize
`Alt+Arrow`	Navigate (no prefix)
`prefix + Space`	Cycle layouts
`prefix + q`	Show pane numbers
`prefix + !`	Pane to window

12.4 Copy Mode

Key	Action
`prefix + v`	Enter copy mode
`v`	Start selection
`y`	Copy selection
`Ctrl+v`	Rectangle select
`/ or ?`	Search fwd / back
`q`	Exit copy mode
`prefix + ]`	Paste

13 Included Scripts Reference

Script	Purpose
`install-tmux.sh`	Install tmux on macOS (Homebrew) or Linux (apt/dnf/pacman)
`setup-tmux-config.sh`	Generate a documented `~/.tmux.conf` with sensible defaults
`setup-tpm.sh`	Install Tmux Plugin Manager and enable plugins
`tmux-sessionizer.sh`	Fast project-based session creation with fzf
`tmux-layout-dev.sh`	Create a 3-window development layout
`tmux-cheatsheet.sh`	Display a quick-reference cheatsheet in the terminal

Table 15: Script inventory

All scripts are in the `scripts/` directory. Make them executable with:

```
chmod +x scripts/*.sh
```

Optionally, copy frequently-used scripts to your PATH:

```
cp scripts/tmux-sessionizer.sh ~/.local/bin/tmux-sessionizer
cp scripts/tmux-cheatsheet.sh ~/.local/bin/tmux-cheatsheet
```