Tmux: Keep a Long Job Running After You Close the Laptop

The fix for work that dies when your laptop sleeps

You SSH into your server and start something that takes a while: a document batch, a scraper pulling county public notices, a database migration, a big file transfer, a model download measured in tens of gigabytes. Then your wifi blinks, or your laptop sleeps, or you have to drive to the courthouse. The connection drops, the job dies with it, and you start over from zero.

Tmux fixes exactly that. It is the first tool I install on any server I run, and it is the difference between a job that survives a dropped connection and a job you have to babysit with the lid open.

What it actually is

Tmux is short for "terminal multiplexer." The plain-English version: it runs your terminal session *on the server* instead of on your laptop. You start a session, do your work, then detach. The session keeps running on the server with nothing attached to it. Reconnect later from any machine, reattach, and your work is exactly where you left it: output, running processes, scroll history, all of it.

That is the whole pitch. Everything below is detail.

Install it

```
# Linux
sudo apt install tmux     # Debian / Ubuntu
sudo dnf install tmux     # Fedora
sudo yum install tmux     # CentOS / RHEL

# macOS (Homebrew)
brew install tmux
```

On Windows you run tmux inside WSL (Windows Subsystem for Linux), or you SSH from Windows Terminal into a Linux box that has it. Tmux is a Linux/Unix tool; it lives on the server, not on Windows itself.

Educational only. Not legal advice. No attorney-client relationship. ©2026 Chariton Media.

The one move that matters: detach and reattach

If you learn nothing else, learn this. It covers ninety percent of why a lawyer running their own server wants tmux.

```
# SSH into the server, then start a named session
tmux new -s batch

# ...run your long job here...

# Detach: the job keeps running on the server
# Press the prefix, then d
Ctrl-b  then  d
```

Now close your laptop, lose your connection, drive across town, whatever. The job is still running on the server. When you are ready:

```
# SSH back in from anywhere, then reattach
tmux attach -t batch
```

You land back in the same session, mid-job, as if you never left. A few supporting commands:

```
tmux ls                          # list your running sessions
tmux kill-session -t batch       # end a session when you are done
tmux capture-pane -pt batch | tail -20   # peek at a job's output WITHOUT
attaching
```

That last one is underrated. You can check on an overnight job from a one-line command without opening the session at all.

Sessions, windows, panes

The vocabulary, fast:

- **Session**: a whole workspace. You can have several, each named (`batch`, `firm`, `scrapers`).
- **Window**: a tab inside a session.
- **Pane**: a split inside a window, so two things share one screen.

Every tmux command starts with a **prefix key**, which is `Ctrl-b` by default. You press `Ctrl-b`, let go, then press the command key.

The keys you actually need

The official list has dozens of bindings. Here are the ones that earn their keep:

Educational only. Not legal advice. No attorney-client relationship. ©2026 Chariton Media.

```
Ctrl-b  c     new window
Ctrl-b  n     next window
Ctrl-b  p     previous window
Ctrl-b  ,     rename the current window
Ctrl-b  &     close the current window
Ctrl-b  [     enter scroll / copy mode (then arrow keys or PgUp; q to quit)
Ctrl-b  x     kill the current pane
Ctrl-b  ?     show every key binding
```

Scroll mode (`Ctrl-b [`) is the one beginners miss. A tmux pane does not scroll with your mouse wheel by default; you enter copy mode first, then scroll back through the output. The config below fixes that if you would rather just use the wheel.

A real example: the overnight job

Say you are running a document-assembly batch or a scraper that pulls a few hundred records and takes an hour. You want it to run whether or not you are connected.

```
ssh you@your-server
tmux new -s nightly
node run-batch.js        # or python scraper.py, or your migration, etc.
# Ctrl-b then d to detach
exit                     # close the SSH session; the job keeps going
```

In the morning:

```
ssh you@your-server
tmux attach -t nightly   # see exactly how it finished
```

No `nohup`, no juggling background processes, no wondering whether the job died when your hotel wifi dropped at 2 a.m.

A small config worth setting

Drop this in `~/.tmux.conf` on the server. It turns on the mouse (scroll and click to select a pane), gives you real scrollback, and makes splitting panes intuitive. Reload it with `tmux source-file ~/.tmux.conf` or just start a new session.

Educational only. Not legal advice. No attorney-client relationship. ©2026 Chariton Media.

```
# ~/.tmux.conf
set -g mouse on                       # wheel scroll + click to select
set -g history-limit 10000            # keep more scrollback
set -g default-terminal "screen-256color"

# split with | and -  (more memorable than the defaults)
bind | split-window -h
bind - split-window -v

# reload config without restarting
bind r source-file ~/.tmux.conf \; display "Config reloaded"
```

Some people also remap the prefix from `Ctrl-b` to `Ctrl-a` because it is easier to reach. That is a personal call; I leave it on `Ctrl-b` so my muscle memory matches every tutorial and every other server I touch.

The part the developer tutorials skip: privilege

Here is the lawyer-specific catch. A detached tmux session is *still alive on the server*. Anyone who can log into that server as your user can run `tmux attach` and see whatever is on that screen, plus scroll back through everything in the buffer. If that buffer holds a client name, a settlement number, or a chunk of a privileged document, it is sitting there in plain text for the next person with access.

So treat tmux like any other place client data lives:

- Run it only on servers **you** control and have locked down. A detached session on a shared or borrowed box is an exposure.
- When you finish sensitive work, do not just detach; `tmux kill-session` so the buffer is gone.
- Be deliberate about what scrolls through a long-lived session in the first place.

This is the same instinct behind running a privilege check before you paste anything into a public AI tool. The session that quietly persists is convenient for you and convenient for anyone else who reaches that machine.

Going further

Two plugins make tmux survive a server reboot, not just a dropped connection:

- **tmux-resurrect**: save and restore your sessions, windows, and panes.
- **tmux-continuum**: save that state automatically on a timer.

They need a small plugin-manager setup, which is past the "memorize four commands" scope of this tip. The full config plus a one-command script that builds a labeled multi-window workspace for you is in the Founders walkthrough.

Educational only. Not legal advice. No attorney-client relationship. ©2026 Chariton Media.

The playbook

Four commands carry the whole thing:

```
tmux new -s NAME       # start
Ctrl-b  d              # detach (job keeps running)
tmux attach -t NAME    # come back
tmux kill-session -t NAME   # end it
```

Memorize those and you stop losing long jobs to a closed laptop.

Educational only. Not legal advice. No attorney-client relationship. ©2026 Chariton Media.