# A pocket terminal for your Linux laptop with ttyd and Tailscale

Ronald G. Thomas

2026-04-15

Figure 1: A smartphone resting screen-up on a wooden desk beside a closed laptop, connected by a visible cable, suggesting a private link between the two devices.

*A browser-based interface to a persistent shell, reachable over an authenticated private network, provides a workable mobile surface for observing and intervening in long-running tasks.*

## Introduction

The ergonomics of operating an SSH client on a mobile device become apparent only when the task is non-trivial. Identity-file selection, modifier-key entry on a soft keyboard, reliable reconnection after the device sleeps, and the rendering of a monospace font at phone or tablet scale each impose non-negligible overhead. In practice these frictions accumulate to the point where many users defer routine oversight work until they regain physical access to the host machine. The resulting delay is not merely inconvenient; it creates an asymmetry between the pace at which background work progresses and the pace at which an operator can respond to it.

Two components, used together, reduce this overhead substantially. The first, `ttyd`, is a small C daemon that serves a terminal emulator over HTTP using the xterm.js JavaScript library; any standards-compliant browser becomes a client without additional software. The second, Tailscale, builds a peer-to-peer WireGuard mesh between enrolled devices and provides an authenticated overlay network, so that the `ttyd` listener is addressable only by devices already admitted to the private tailnet rather than by arbitrary hosts on the public internet. The combination yields a single URL that is reachable from an enrolled phone, is invisible to every other network the laptop is attached to, and resumes a persistent tmux session on each visit.

We document the configuration end to end. The coverage includes installation of `ttyd`, Tailscale, and tmux on an Ubuntu 24.04 laptop, binding of the `ttyd` listener to the Tailscale network interface, and wrapping of the service in a per-user systemd unit that survives logout and reboot. It also covers client-side configuration on iOS and optional hardening with a TLS certificate issued by `tailscale cert`. Appendices cover a sample work session and the full teardown procedure. Corrections and alternative approaches are welcome.

More formally, we document here an entry point to the Remote Access concern of the Workflow Construct described in A Workflow Construct for the Modern Data Scientist. Tailscale plus `ttyd` is one specific solution to the cross-machine, cross-network reachability problem; the broader Remote Access family also includes `mosh` for high-latency-link resilience and standard SSH for the unauthenticated-network case. The present post is the browser-shell-on-mobile leaf of that family.

## Motivations

The configuration described here was motivated by the following requirements, each tied to a concrete limitation of the alternatives:

- Monitoring and occasional intervention in long-running interactive sessions, including agentic coding workflows that may run for hours, without requiring physical access to the host machine.
- Avoidance of identity-file management and modifier-key entry on a small touchscreen, which are the two principal ergonomic costs of native SSH clients on iOS.
- Elimination of any public-internet exposure on the laptop. Opening inbound port 22 through a home router, combined with dynamic DNS, would grant global addressability to a device that regularly attaches to untrusted networks; this is an unacceptable risk profile for a personal workstation.
- Network-location independence. A single stable URL should be usable from the home network, the institutional network, and a cellular carrier's network without client-side reconfiguration.
- Generalization to future deployments. The component choices (a WireGuard-based overlay, an HTTP-native terminal, a per-user systemd unit) transfer cleanly to a cloud-hosted VPS workstation, which was identified as a plausible next step.

## Objectives

The scope of this post is the following set of verifiable deliverables:

1. Install `ttyd`, Tailscale, and `tmux` on a Linux laptop and confirm each by version query.

2. Run `ttyd` bound exclusively to the Tailscale network interface, protected by HTTP basic authentication, and attached to a persistent tmux session.
3. Wrap the service in a per-user systemd unit that remains active across logout and reboot.
4. Connect to the service from an iPhone browser over the tailnet, verify that interactive input and output behave correctly, and document a complete teardown procedure.

The configuration is presented as a reproducible reference rather than a tutorial. Each command and configuration directive is annotated with the rationale for its inclusion.

Figure 2: A small brass padlock resting on a folded paper map with a single dotted line drawn between two marked points.

## What is ttyd?

`ttyd` is a small C daemon, approximately six thousand lines of source, that serves a terminal emulator over HTTP. On the server side it allocates a pseudo-terminal (pty), spawns a process attached to it (commonly a shell or a multiplexer such as tmux), and exposes the pty's input and output through a WebSocket on an HTTP listener. On the client side the browser loads the xterm.js library, opens the WebSocket, and renders a full VT100-family terminal in a `<canvas>` element; keystrokes are forwarded as they are generated, and output bytes are written back to the terminal buffer.

In functional terms the role of `ttyd` is analogous to that of an SSH daemon, but the transport is HTTP(S) and the client is a browser rather than a native terminal emulator. This has three practical consequences. First, no client-side software installation is required; any device with a recent browser is a viable client. Second, the authentication and transport-security story is different: `ttyd` itself provides only HTTP basic authentication, and is not intended to be exposed to untrusted networks without an enclosing security layer (a reverse proxy with client-certificate authentication,

or in this configuration, a Tailscale overlay). Third, session persistence across client disconnections is delegated: closing the browser tab tears down the WebSocket, but the pty and its attached process continue to run as long as `ttyd` is running, which is why this configuration always attaches `ttyd` to a tmux session rather than to a bare shell.

As a minimal example, `ttyd -W -p 7681 zsh` starts a writable shell on port 7681; any host that can establish a TCP connection to that endpoint obtains an interactive terminal. The Tailscale layer constrains the meaning of 'any host that can reach that endpoint' to devices that have been authenticated into the tailnet, and routes the traffic over an encrypted WireGuard tunnel rather than over a TCP connection that transits intermediate networks in clear text.

## Architecture Overview

The runtime system is best understood as three layered components, each with a distinct responsibility:

1. **The terminal process.** tmux runs as a long-lived user process on the laptop, maintaining one or more pty pairs and the associated shell state. Its lifecycle is independent of any network connection and outlasts both the `ttyd` daemon and the browser.
2. **The HTTP gateway.** `ttyd` runs as a systemd user service, attaches to the tmux session specified on its command line, and serves the terminal contents over HTTP. It listens only on the Tailscale virtual interface (`tailscale0`), so its socket is unreachable from the physical wired, wireless, or cellular interfaces of the host.
3. **The transport overlay.** Tailscale runs as a system-level daemon (`tailscaled`) and maintains the `tailscale0` WireGuard interface. All packets arriving on that interface have already been authenticated against the tailnet's public-key infrastructure and decrypted from their WireGuard envelopes. On the client (the phone), the Tailscale iOS app maintains the corresponding peer relationship and presents the tailnet as a routeable network.

The request path for a single keystroke therefore traverses: browser on phone, iOS Tailscale VPN, WireGuard tunnel (direct peer-to-peer where possible, or via a DERP relay when both endpoints are behind restrictive NATs), `tailscale0` on the laptop, `ttyd` listener on port 7681, the pty master owned by `ttyd`, and finally the tmux process that is the pty slave. The response path returns bytes by the reverse route. All segments between the phone and the laptop are encrypted by WireGuard; the segment between `ttyd` and `tailscale0` is local to the loopback-like virtual interface and does not leave the host.

## Security Model

The configuration is designed to withstand four threats, listed here in approximate order of likelihood: (a) an attacker scanning the public IP addresses of the laptop's current network for open ports, and (b) an attacker on the same Layer-2 network as the laptop (for example, a shared coffee-shop SSID) attempting to observe or connect to local services. The remaining two are (c) an attacker who compromises a single device on the tailnet and attempts to move laterally, and (d) an attacker with access to the basic-authentication password but not to the tailnet.

Three layered controls address these threats:

- **Network-layer authentication (Tailscale).** Only devices that have completed the OAuth-style authentication flow with the tailnet's coordination server and received a signed node key can establish WireGuard sessions with the laptop. This eliminates threats (a) and (b): the `ttyd` port is not even visible to hosts outside the tailnet. An attacker on a shared SSID sees only the encrypted WireGuard packets, which are indistinguishable from other UDP traffic.
- **Interface binding (`ttyd -i tailscale0`).** The listener is bound to the Tailscale virtual interface rather than to `0.0.0.0`. If the tailnet were somehow compromised at the coordination-server level, or if Tailscale itself were misconfigured to route traffic from unexpected sources into `tailscale0`, the service would still be unreachable from the physical interfaces. This is a defense-in-depth control.
- **Application-layer authentication (HTTP basic auth).** A username and password are required to open a WebSocket, providing a final barrier against threat (c): an attacker who has compromised one tailnet device cannot open a shell on the laptop without also possessing the `ttyd` credentials. Basic authentication is not a strong control in isolation (the credentials are sent on every request and are visible in `ps` output on the host), but it is appropriate as the innermost layer of a multi-layer model.

Threat (d) is out of scope: an attacker with the basic-auth password but no tailnet credentials has no routable path to the service. This is the intended shape of the model: Tailscale provides confidentiality and network-layer authentication; basic authentication provides an independent defense against compromised tailnet peers; the interface binding ensures that neither of the other two controls is single-point-of-failure.

Appendix A describes an optional additional layer (TLS via `tailscale cert`) that closes the remaining residual risk of plaintext credentials on the wire between `ttyd` and the `tailscale0` interface, at a modest operational cost.

## Prerequisites

The configuration assumes the following environment:

- **Operating system:** Ubuntu 24.04 on the laptop acting as the server. Other Linux distributions can be accommodated with minor substitutions in the package-manager invocations.
- **Client device:** an iPhone running iOS 17 or later with Safari as the browser. Android and desktop browsers function identically with respect to the xterm.js client.
- **Hardware:** any laptop with sufficient resources to run Ubuntu; the runtime footprint of the services described here is negligible.
- **Prior installation state:** a Linux user account with `sudo` privileges, and a Tailscale account (the free tier is adequate for the use cases described here).
- **Prior knowledge:** familiarity with systemd unit-file syntax and general Linux administration at the shell. Prior experience with tmux is helpful but not required.
- **Time investment:** approximately thirty minutes for installation, configuration, and verification of the first successful browser connection.

Where any assumption fails to hold, the commands below may require adaptation.

## Installation

On Ubuntu, the three required packages are available from the distribution repositories and the Tailscale install script:

```
sudo apt update
sudo apt install -y ttyd tmux
curl -fsSL https://tailscale.com/install.sh | sh
sudo tailscale up
```

The `tailscale up` command prints an authentication URL. Open it in any browser, sign in, and confirm the device appears in the tailnet.

> **Note**
>
> If the distribution ships a `ttyd` older than 1.7, install a prebuilt binary from the ttyd releases page instead. Earlier versions lack the `-i` interface-binding flag used below.

Confirm the installs succeeded:

```
ttyd --version        # Expected: ttyd version 1.7.x
tailscale version     # Expected: 1.64.x or later
tmux -V               # Expected: tmux 3.4 or later
tailscale ip -4       # Prints the laptop's 100.x.y.z address
```

## Configuration

Two configuration artifacts drive the setup: a single `ttyd` command line and a systemd user unit that wraps it. Both are stored in full under `analysis/configs/` in the post repository.

### The ttyd invocation

```
ttyd -W \
  -p 7681 \
  -i tailscale0 \
  -c user:'REPLACE_WITH_STRONG_PASSWORD' \
  tmux new -A -s main
```

Rationale for each flag:

- `-W` enables write input on the pty. Without this flag the browser session is read-only, which is not useful for the interactive work that motivates the configuration.
- `-p 7681` specifies the TCP listening port. Any unused port is acceptable; 7681 is the `ttyd` project default and is used here for consistency with upstream documentation.

- `-i tailscale0` binds the listener exclusively to the Tailscale virtual interface. The service is therefore unreachable from physical wired, wireless, or cellular interfaces of the laptop, regardless of which network the laptop is attached to at any given moment. The interface name should be confirmed with `ip -br link`, as it is nominally fixed but may differ on non-default configurations.
- `-c user:password` enables HTTP basic authentication. The credentials are an application-layer defense-in-depth measure; the primary access control is the Tailscale interface binding, and the basic-auth credentials are an independent safeguard against the scenario in which a tailnet peer is itself compromised.
- `tmux new -A -s main` invokes tmux in a mode that creates the session if it does not exist and attaches to it otherwise. This form is idempotent, which is essential inside a systemd `ExecStart` directive. It also ensures that every browser connection attaches to the same session, so that shell state persists across client disconnections.

### The systemd user unit

Manual launches do not survive logout. The unit below, saved as `~/.config/systemd/user/ttyd.service`, runs the service as the logged-in user and restarts on failure.

```
# ~/.config/systemd/user/ttyd.service
# Tested on ttyd 1.7.7, Ubuntu 24.04
[Unit]
Description=ttyd terminal over Tailscale
After=network-online.target tailscaled.service
Wants=network-online.target

[Service]
Type=simple
ExecStart=/usr/bin/ttyd -W -p 7681 -i tailscale0 \
  -c user:REPLACE_WITH_STRONG_PASSWORD \
  /usr/bin/tmux new -A -s main
Restart=on-failure
RestartSec=5

[Install]
WantedBy=default.target
```

Enable and start:

```
systemctl --user daemon-reload
systemctl --user enable --now ttyd.service
systemctl --user status ttyd.service
```

To allow the user service to start before the user logs in at the console (so the laptop is reachable after an unattended reboot):

```
sudo loginctl enable-linger "$USER"
```

Restrict the unit file permissions so the basic-auth password is not world-readable:

```
chmod 600 ~/.config/systemd/user/ttyd.service
```

Figure 3: A pocket-sized leather notebook lying open next to a laptop, both showing the same short handwritten list, suggesting mirrored, synchronized content.

## Verification

Three checks confirm the stack is operating correctly.

```
# 1. Service is running
systemctl --user is-active ttyd.service
# Expected: active

# 2. Port is bound to the Tailscale interface only
ss -ltnp | grep 7681
# Expected: LISTEN on 100.x.y.z:7681 (tailnet address),
# NOT on 0.0.0.0:7681

# 3. End-to-end reachability from a second tailnet device
curl -u user:PASSWORD -I \
  http://$(tailscale ip -4):7681
# Expected: HTTP/1.1 200 OK
```

If step 2 shows `0.0.0.0:7681`, the `-i tailscale0` flag is not taking effect; the service is exposed to every network the laptop joins and must not be started until fixed.

## Connecting from the iPhone

The client-side procedure consists of four steps:

1. Install the Tailscale application from the App Store.
2. Authenticate with the same account used on the laptop and confirm that the laptop appears under **Devices** in the Tailscale client.
3. Enable the Tailscale VPN toggle. iOS will report an active VPN connection in the status bar.
4. In Safari, navigate to the service URL:

   ```
   http://laptop.tailnet-name.ts.net:7681
   ```

   The MagicDNS name is reported by `tailscale status` on the laptop. The bare-IP form (`http://100.x.y.z:7681`) is functionally equivalent and is useful for diagnostic purposes when MagicDNS resolution is suspected to be failing.
5. Supply the basic-authentication credentials. The browser will render an interactive terminal attached to the `main` tmux session.

> **Tip**
>
> Adding the URL to the iPhone home screen via Safari's **Share > Add to Home Screen** creates a standalone web-application icon. When launched from that icon, the page runs in a full-screen container without browser chrome, which improves usable display area on a small screen.

## Daily Workflow

The table below summarizes the commands and gestures that are most frequently required during routine use:

| Command or gesture | Action |
|---|---|
| Open home-screen icon | Attach to `main` tmux session |
| `Ctrl-b d` (within tmux) | Detach without terminating the session |
| `Ctrl-b c` | Create a new window within the session |
| `Ctrl-b n` / `Ctrl-b p` | Cycle to next / previous tmux window |
| `systemctl --user restart ttyd.service` | Restart the terminal service |
| `journalctl --user -u ttyd.service -f` | Follow the service log in real time |
| `tailscale status` | List reachable devices on the tailnet |
| Pinch-zoom in Safari | Adjust terminal font size |

A Bluetooth keyboard paired with the iPhone substantially improves the input ergonomics, to the point that the configuration becomes suitable for sustained interactive work rather than brief observation. Safari forwards the `Ctrl`, `Esc`, `Tab`, and arrow keys to xterm.js without rebinding. A keyboard on which Caps Lock has been remapped to `Ctrl` (an iOS system setting) reduces hand-position strain during extended tmux use.

## Operational Considerations

The following issues arise sufficiently often during initial deployment and routine use to warrant explicit documentation. Each is presented with its symptom and its remediation.

1. **Incorrect interface binding.** If `ss -ltnp` shows `0.0.0.0:7681` rather than the tailnet address, the `-i tailscale0` directive has not taken effect. The interface name should be confirmed with `ip -br link`, as it may differ on systems with non-default Tailscale installations or with systemd-networkd renaming. The service should not be left running until the binding has been verified to be correct, because a listener on `0.0.0.0` is reachable from every network the laptop attaches to.

2. **Lid-close suspension.** A laptop whose power-management policy suspends the system on lid closure is not reachable over the network while suspended. The remediation is one of: physically maintain an open lid; modify the desktop environment's lid-close behavior to 'do nothing'; or, for transient tasks, wrap the command with `systemd-inhibit --what=sleep` to prevent suspension for the duration of the inhibition.

3. **Loss of session persistence.** If closing the browser terminates the shell, the `ExecStart` directive is launching a bare shell directly rather than attaching to a tmux session. Only the `tmux new -A -s main` form, with tmux as the terminal's attached process, delivers persistence across client disconnections.

4. **Service termination at logout.** Without `loginctl enable-linger <user>`, systemd user services are terminated when the user's last interactive session ends. Lingering must be enabled once per user and, being a system-level setting, requires `sudo`.

5. **Plaintext credentials in the unit file.** The basic-authentication password is stored in plaintext in the `ExecStart` line of the unit file. The unit permissions must be restricted with `chmod 600`, and the unit file must not be committed to a shared repository. For higher sensitivity requirements, use a systemd credentials file (`LoadCredential=`) or migrate to Tailscale Serve as described under Opportunities for Improvement.

6. **MagicDNS resolution failures.** Tailnet hostnames must include the full tailnet suffix (for example, `laptop.tailnet-name.ts.net`). A bare hostname resolves only if MagicDNS is enabled in the Tailscale admin console and if the client has accepted the DNS configuration pushed by the Tailscale daemon; clients that predate the MagicDNS enablement may require a Tailscale toggle cycle to refresh.

7. **iOS input autocorrection.** Safari on iOS auto-capitalizes the first character entered into a text field by default, which can corrupt the first character of a password or a command. The appropriate remediations are to disable auto-capitalization under **Settings > General > Keyboard**, or to use a Bluetooth keyboard, which is not subject to the on-screen keyboard's input transformations.

## Uninstall / Rollback

To remove the setup, disable the service and uninstall the packages.

```
# 1. Stop and disable the service
systemctl --user disable --now ttyd.service
rm ~/.config/systemd/user/ttyd.service
systemctl --user daemon-reload

# 2. Disable lingering if no other user service needs it
sudo loginctl disable-linger "$USER"

# 3. Uninstall packages
sudo apt remove --purge -y ttyd
# Leave tmux and tailscale if used elsewhere; otherwise:
sudo apt remove --purge -y tmux
sudo tailscale down
sudo apt remove --purge -y tailscale
```

Remove the laptop from the tailnet via the Tailscale admin console if the device is being retired.

Figure 4: A closed laptop on a desk in dim evening light, with a smartphone glowing softly nearby, suggesting an unattended session still quietly running.

# Discussion

## Observations

### Conceptual observations:

- A browser is an adequate terminal client when the network transport is handled by a component specifically engineered for authenticated overlay networking. The SSH daemon is one of several defensible approaches to remote shell access; in this configuration the transport responsibility is relocated from SSH to WireGuard, and the authentication responsibility is distributed across WireGuard, `ttyd` basic auth, and the interface binding.
- MagicDNS eliminates the operational burden of dynamic DNS, port forwarding, and firewall-rule maintenance for personal-scale remote access. The trade-off is dependence on the Tailscale coordination server for name resolution.
- The interface-binding flag on `ttyd` is the single most consequential hardening decision in the configuration. A service bound to `0.0.0.0` with HTTP basic authentication represents a qualitatively different risk profile from a service bound to a WireGuard interface; the former is subject to public-internet scanning, the latter is not.
- Per-user systemd units, combined with `loginctl enable-linger`, provide a clean abstraction for background services that require user-level file system access without requiring system-level privileges beyond the initial linger enablement.

### Technical observations:

- The `tmux new -A -s <name>` form is both a creator and an attacher, and is therefore idempotent. This makes it safe to invoke from an `ExecStart` directive, where a non-idempotent invocation would cause a restart loop.
- The `ss -ltnp` command is more direct and more precise than `netstat -ltnp` for determining the interface on which a service is listening, and it is the standard utility on distributions that have replaced `net-tools` with `iproute2`.
- The `tailscale cert` command issues short-lived TLS certificates for MagicDNS hostnames, enabling end-to-end HTTPS without involvement of a public certificate authority. Certificates expire and must be reissued on a schedule, as discussed in Appendix A.
- The Safari **Add to Home Screen** action produces a standalone web-application surface that removes browser chrome and recovers screen area, which materially improves usability on any URL that is accessed with regularity.

### Recurring Pitfalls:

- Omission of the `-W` flag yields a read-only terminal, which presents as a non-interactive first connection and is easily misdiagnosed as a network or authentication problem.
- Binding to `tailscale0` without first confirming the interface name (via `ip -br link`) produces a service that fails to start and logs an interface-not-found error; the service is inert but not obviously so on casual inspection.
- The basic-authentication password is visible both in `ps` output on the host and in the unit file. It should be classified as a moderate-sensitivity secret: rotated on suspicion of exposure, but not requiring the operational controls appropriate to a high-sensitivity credential such as an SSH private key.

- iOS aggressively reclaims memory from backgrounded Safari tabs. The WebSocket reconnects transparently on tab reactivation, but any command-line input that had been typed but not submitted at the time of backgrounding is lost.

### Limitations

The configuration has several explicit constraints that should be understood at the outset:

- **Single-user scope.** The unit file and basic-authentication credentials are per-user. Multi-user access requires either parallel service instances on distinct ports or a front-end reverse proxy that demultiplexes on authenticated identity.
- **Absence of audit logging.** `ttyd` does not log the commands executed within the pty. Where accountability or forensic capability is required, a separate shell-level or system-level auditing facility (for example, `auditd` with appropriate rules) must be configured.
- **Input ergonomics on a phone-scale device.** Even with a Bluetooth keyboard and the small-screen optimizations, extended authorship of new code on a handheld device remains slower than on a conventional workstation. The configuration is well suited to monitoring, intervention, and short modifications; it is not a substitute for a primary development environment.
- **Dependence on Tailscale infrastructure.** New connections require reachability to the Tailscale coordination server for key exchange and peer discovery. Existing WireGuard sessions are unaffected by a coordination-server outage, but reconnection after a client restart requires coordination-server availability.
- **Absence of session recording.** The configuration does not record session input or output. Tools such as `asciinema`, `tmux-logging`, or `script(1)` can be added if session provenance is required.

### Opportunities for Improvement

Several extensions would strengthen the configuration without departing from its overall design:

1. Adopt end-to-end TLS using `tailscale cert`, so that an accidental mis-binding of `ttyd` to a non-Tailscale interface would not expose an unencrypted session. Appendix A documents the procedure.
2. Replace HTTP basic authentication with Tailscale Serve or Tailscale Funnel, both of which delegate authentication to the tailnet identity of the client. This removes the need for a shared password and eliminates the associated secret-management burden.
3. Deploy the same configuration on an always-on VPS reached through Tailscale. The benefit is session durability across laptop sleep or shutdown; the cost is the VPS itself and the slightly more involved administrative story.
4. Instantiate a second, read-only `ttyd` on a distinct port that runs a scoped command such as `htop` or a log-tailing command. This provides an observation-only surface suitable for short status checks without opening the full shell.
5. Curate a phone-optimized tmux configuration (`tmux.conf.phone`) with shortened status-bar labels and fewer segments, and load it conditionally when `ttyd` detects a small-screen client.
6. Encapsulate the installation in an idempotent shell script under `analysis/configs/`, suitable for provisioning identical configurations on additional machines.

## Conclusion

A terminal served over HTTP and reached through a private WireGuard mesh constitutes a workable mobile surface for a laptop that may be closed, stationed in another room, or physically distant from the operator. The marginal cost is modest: three package installations, a single systemd user unit, and an initial configuration effort of approximately thirty minutes. The benefit is that long-running work on the laptop, including interactive agentic coding sessions that may run for several hours, remains observable and steerable without any exposure of the laptop to the public internet, without dynamic DNS infrastructure, and without reliance on a native SSH client on the mobile device.

The principal design insight is that a browser is an adequate terminal client when the transport-layer authentication and confidentiality problems are delegated to a component engineered specifically for that purpose. Once Tailscale and `ttyd` are in place, the same configuration generalizes to a cloud-hosted VPS: only the server endpoint changes, and the tailnet membership, the `ttyd` invocation, and the client-side procedure are invariant.

In conclusion, four points merit emphasis. First, the single most consequential line in the configuration is `-i tailscale0` on the `ttyd` invocation: that binding decision removes the service from every non-tailnet interface and makes the rest of the security model meaningful. Second, a per-user systemd unit combined with `loginctl enable-linger` provides a logout-safe and reboot-safe service without any system-level privileges beyond the initial `linger` enablement. Third, the `tmux new -A -s main` argument inside the `ExecStart` directive is the mechanism that transforms an ephemeral browser tab into a durable working session. Fourth, the Safari 'Add to Home Screen' action yields a standalone web-application presentation that recovers enough screen area to make the terminal practically usable on phone-scale displays.

## See Also

**Related posts:**

- Provisioning AWS EC2 Instances: a similar setup post for a cloud workstation that pairs naturally with the configuration here.

**Key resources:**

- ttyd project page
- Tailscale installation guide
- Tailscale MagicDNS
- tailscale cert documentation
- tmux manual
- systemd user units

---

## Extended Glossary

The following terms appear throughout the post. Definitions are compact and operational, prioritizing the usage relevant to this configuration over full generality.

**ACL (access control list).** In Tailscale, a declarative policy document that restricts which tailnet members may reach which services on which ports. Default policy is permissive; explicit ACLs tighten the model.

**Basic authentication (HTTP).** The authentication scheme defined by RFC 7617 in which the client sends a header of the form `Authorization: Basic base64(user:password)` on each request. Sufficient when transport-layer confidentiality is assured by another mechanism; inadequate alone over plaintext HTTP.

**Coordination server (Tailscale).** The central component of the Tailscale control plane, responsible for device authentication, public-key distribution, and ACL enforcement. It does not carry data traffic; user traffic flows peer-to-peer over WireGuard or via DERP relays.

**DERP (Designated Encrypted Relay for Packets).** Tailscale-operated relay servers used when two peers cannot establish a direct WireGuard connection due to symmetric or restrictive NAT. Traffic through a DERP relay remains end-to-end encrypted; the relay cannot read it.

**Dynamic DNS.** A service that updates a DNS A or AAAA record in response to changes in an endpoint's public IP address. Historically used for home-hosted services but requires an open public port on the host. This configuration deliberately avoids it in favor of MagicDNS over Tailscale.

**ExecStart.** A systemd unit directive that specifies the command line to run when the service is started. Must be an absolute path with no shell interpretation; metacharacters are passed literally to the process.

**HTTP.** The Hypertext Transfer Protocol, as used here in its unencrypted form over a Tailscale-encrypted transport. The `ttyd` listener speaks HTTP, and the security model delegates confidentiality to the WireGuard layer below it.

**HTTPS.** HTTP over TLS. In this configuration, HTTPS is optional (Appendix A) because the transport is already authenticated and encrypted by Tailscale; adding HTTPS provides defense in depth against an accidental mis-binding of the listener.

**Interface binding.** The act of restricting a listening socket to accept connections only on a named network interface, typically by specifying an interface name or an IP address belonging to that interface. Here, `ttyd -i tailscale0` binds the listener to the Tailscale virtual interface.

**Linger (systemd).** A setting, enabled per user by `loginctl enable-linger <user>`, that causes systemd to keep a user's services running even when no session is active on that user's behalf. Required for a systemd user unit to persist across logout and reboot.

**MagicDNS.** A Tailscale feature that automatically resolves hostnames of the form `<device-name>.<tailnet-name>.` to tailnet IP addresses, with the resolution delivered via the tailnet VPN rather than the system resolver. Eliminates the need for dynamic DNS or manual hosts-file maintenance.

**NAT (network address translation).** The technique by which a router rewrites source or destination addresses in IP packets, commonly used to share a single public IPv4 address among

many private hosts. WireGuard and DERP together are responsible for establishing communication across NATs without requiring port forwarding.

**NAT traversal.** The set of techniques used to establish peer-to-peer connectivity between hosts behind NAT devices, including STUN-style discovery and UDP hole punching. Tailscale performs NAT traversal automatically; DERP is the fallback when it fails.

**Overlay network.** A network built logically on top of an underlying physical network, with its own address space and its own forwarding rules. The Tailscale `100.x.y.z/10` address space is an overlay over the underlying internet.

**Posit Package Manager.** A binary R package mirror (formerly RStudio Package Manager) used by the zzcollab Docker image to accelerate package installation. Not directly relevant to the ttyd configuration, but present in the repository's reproducibility infrastructure.

**Pseudo-terminal (pty).** A pair of character-device endpoints in the kernel, the master and the slave, that emulates a serial line between a terminal program and a process. `ttyd` owns the master end; tmux (or another attached process) reads from and writes to the slave end.

**Quarto.** The open-source scientific and technical publishing system used to render this post. Produces HTML, PDF, and other formats from a unified source document. Relevant to the blog infrastructure rather than to the ttyd setup.

**renv.** The R package-version pinning and isolation tool used throughout the zzcollab research framework. Not directly used by the ttyd configuration.

**Rocker.** A family of Docker images providing R and common toolchains; `rocker/tidyverse` is the base image for this blog's rendering environment. Again, infrastructure rather than subject matter.

**Session (tmux).** A named, persistent grouping of tmux windows and panes that survives detachment from any client. The `main` session used here is created by `tmux new -A -s main` and is attached by all subsequent `ttyd` connections.

**ss.** A socket-statistics utility from the `iproute2` suite, preferred over the legacy `netstat` tool for listing listening sockets. The invocation `ss -ltnp` lists TCP listeners with their owning process and the bound interface.

**SSH (Secure Shell).** A cryptographic network protocol for remote shell access, defined principally by RFC 4251 through RFC 4254. This configuration deliberately does not use SSH for inbound access, although SSH may still be used for other purposes on the laptop.

**systemd.** The init system and service manager used by most contemporary Linux distributions. Responsible for launching, supervising, and logging long-running services.

**systemd user unit.** A service definition stored under `~/.config/systemd/user/` and managed with `systemctl --user`. Runs as the user rather than as root; requires `loginctl enable-linger` to persist across logout.

**Tailnet.** A private network of devices enrolled in a single Tailscale account. Members are mutually addressable; non-members are not routable to tailnet addresses.

**Tailscale.** A commercial product that builds a peer-to-peer WireGuard mesh between enrolled devices, coupled with a coordination server for authentication and key distribution. The free tier is adequate for personal use.

**Tailscale Funnel.** An optional Tailscale feature that exposes a selected tailnet service to the public internet through a Tailscale-operated reverse proxy, with TLS termination and public-DNS resolution. Not used in this configuration.

**Tailscale Serve.** An optional Tailscale feature that provides TLS termination and routing for tailnet-internal HTTPS services, using certificates issued automatically via `tailscale cert`. A plausible future replacement for the HTTP + basic-auth configuration described here.

**TLS (Transport Layer Security).** The successor to SSL, and the protocol used to encrypt HTTP into HTTPS. Optional in this configuration because Tailscale already provides confidentiality at the WireGuard layer.

**tmux.** A terminal multiplexer that maintains persistent sessions containing windows and panes. Essential to this configuration because it decouples the shell lifetime from the lifetime of any individual client connection.

**ttyd.** The daemon that is the subject of this post. Serves a terminal over HTTP via WebSocket, using xterm.js on the client side.

**WebSocket.** A protocol (RFC 6455) that upgrades an HTTP connection to a full-duplex, persistent framed-TCP channel, used by `ttyd` to stream keystrokes and output between the browser and the pty.

**WireGuard.** A modern encrypted tunnel protocol, noted for its small codebase and its use of Curve25519 and ChaCha20-Poly1305. Tailscale builds its overlay on WireGuard but adds a coordination plane, NAT traversal, and identity management on top.

**xterm.js.** A JavaScript terminal emulator library that implements VT100/VT220/xterm escape-sequence handling in the browser. Rendered by `ttyd` on every client connection.

---

## Reproducibility

**Tested configuration:**

| Component | Version |
|---|---|
| Operating system | Ubuntu 24.04 |
| ttyd | 1.7.7 |
| Tailscale | 1.64.2 |
| tmux | 3.4 |
| Shell | zsh 5.9 |
| Client | iOS 17.4, Safari |
| Last verified | 2026-04-15 |

**Configuration files:**

- `analysis/configs/ttyd.service`: the full systemd user unit

- `analysis/configs/install.sh`: an idempotent install script covering `ttyd`, `tmux`, and Tailscale
- `analysis/configs/tmux.conf.phone`: an optional tmux configuration tuned for small screens

**To reproduce end-to-end:**

```
bash analysis/configs/install.sh
sudo tailscale up
install -m 600 analysis/configs/ttyd.service \
  ~/.config/systemd/user/ttyd.service
# edit the unit and substitute a real password
systemctl --user daemon-reload
systemctl --user enable --now ttyd.service
sudo loginctl enable-linger "$USER"
```

---

## Appendix A: TLS with tailscale cert

For transport encryption end-to-end (rather than relying on Tailscale's WireGuard tunnel alone), issue a certificate for the MagicDNS name and point `ttyd` at it.

```
sudo mkdir -p /etc/ttyd
cd /etc/ttyd
sudo tailscale cert laptop.tailnet-name.ts.net
```

Adjust the unit's `ExecStart` to include:

```
--ssl \
--ssl-cert /etc/ttyd/laptop.tailnet-name.ts.net.crt \
--ssl-key  /etc/ttyd/laptop.tailnet-name.ts.net.key \
```

Access the service at `https://laptop.tailnet-name.ts.net:7681`. Certificates issued by `tailscale cert` expire; schedule a weekly systemd timer to reissue and reload:

```
systemctl --user edit --force --full ttyd-cert-renew.timer
```

## Appendix B: Sample Work Session

The following sequence illustrates a representative end-to-end session, under the assumption that the configuration above is complete and the laptop is running.

**Step 1.** At the workstation, open a local terminal and attach to the shared tmux session:

```
tmux attach -t main
```

**Step 2.** Initiate a long-running interactive task (for example, an agentic coding session or a computation that will run for an extended period). Detach from the tmux session with `Ctrl-b d`. The task continues to execute.

**Step 3.** Depart from the workstation. The laptop remains powered and attached to a network. From any enrolled tailnet device, launch the home-screen icon for the `ttyd` URL and authenticate with the configured credentials. The browser attaches to the same tmux session, and the task's current output appears.

**Step 4.** Observe the output, issue intervention commands as required, and close the browser tab when finished. Closing the tab terminates only the WebSocket connection to `ttyd`; the tmux session and its subprocesses continue to run.

**Step 5.** On return to the workstation, reattach locally with `tmux attach -t main`. The session state is continuous across the entire sequence; no work is lost at any transition.

## Appendix C: Teardown (Undo)

Full removal in order:

1. `systemctl --user disable --now ttyd.service`
2. `rm ~/.config/systemd/user/ttyd.service`
3. `sudo loginctl disable-linger "$USER"` (if no other user service needs it)
4. `sudo apt remove --purge -y ttyd tmux`
5. `sudo tailscale down`
6. `sudo apt remove --purge -y tailscale`
7. Remove the device from the Tailscale admin console.
8. Delete any issued certificates under `/etc/ttyd/`.

> ⚠ Warning
>
> If the laptop is being decommissioned rather than just retired from this workflow, revoke its Tailscale auth key in the admin console. A device removed only locally can reappear on the tailnet if the machine is recovered.

---

## Contact

*Corrections, alternative approaches, and substantive discussion are welcome.*


- **GitHub:** rgt47
- **Twitter/X:** @rgt47
- **LinkedIn:** Ronald Glenn Thomas

- **Email:** Contact form

Feedback is particularly useful on the following topics:

- Corrections or improvements to any of the configuration or the associated reasoning.
- Client-side experience on platforms other than iOS (Android, Chrome OS, iPadOS tablet use).
- Extensions of the pattern, for example Tailscale Serve, Tailscale Funnel, mutual TLS, or certificate-based client authentication, and the operational trade-offs encountered.
- Reports of deployments of this configuration on a cloud VPS rather than on a mobile laptop.

---

## Related posts

This post is not part of the core onboarding sequence: its scenario (a single Ubuntu laptop plus a phone browser) is a niche, advanced remote-access convenience, not something needed to get a new lab member's primary machine productive. It belongs to a separate, smaller *Workflow Extensions* grouping alongside LLM-Augmented Editing for the Workflow Construct.

**Core onboarding series** (*Workflow Construct*), for readers who have not yet completed the base setup:

1. Post 15: A Workflow Construct for the Modern Data Scientist
2. Post 16: Unix Command-Line Workspace Setup for Data Science
3. Post 17: Multi-Laptop macOS Bootstrap
4. Post 18: Setting Up Git for Data Science Workflows
5. Post 19: Setting Up Neovim as a Data Science IDE
6. Post 21: Modern CLI Replacements for the Shell Layer
7. Post 25: Install Linux Mint on a MacBook Air