One Jump Box, Many Users: Multi-User RDP on Ubuntu 26.04

September 3, 2026 / Gavin Jackson

homelab linux ubuntu security essential-eight

I have reached the point in my home lab where the number of things I administer has outgrown the number of places I administer them from. There are servers, a couple of network appliances, the Proxmox host, and an assortment of Raspberry Pis doing jobs I have half forgotten about. Every one of them gets managed from whatever laptop happens to be in reach, with whatever SSH keys and saved credentials happen to be on it. That is convenient, and it is also exactly the kind of sprawl that security frameworks have been warning us about for years.

So this is the first of two articles about building a proper jump host: a single, hardened Ubuntu 26.04 Workstation that multiple users can remote into over RDP, and from which all administration happens. This part covers why you would bother, and how to get multi-user xrdp working on a distribution that has gone all-in on Wayland. Part 2 will put Apache Guacamole in front of it so the whole thing is reachable from a web browser.

Why a Jump Host?

A jump host (also called a jump box, bastion host, or pivot server) is a hardened machine that sits between your users and the things they administer. Instead of every admin connecting to every server from their own workstation, everyone connects to the jump host first, and administers from there.

The Australian Cyber Security Centre's Essential Eight does not use the words "jump host" in the headline mitigation strategies, but the concept is baked into the **restrict administrative privileges** pillar and the supporting ISM guidance. As maturity levels climb, the ACSC pushes you towards a model where:

- Privileged and unprivileged operating environments are separated. Your everyday workstation, with its web browsing and email, is not the same environment you administer production from.
- Administrative access happens from hardened, controlled systems, not from wherever is convenient.
- Privileged credentials are not floating around on general-purpose machines.

A jump host is the practical answer to all three. It gives you a single choke point that you can patch first, harden properly, log exhaustively, and fence off with network rules so that administrative traffic to your servers is only allowed from one address. If a laptop gets lost or compromised, the blast radius is the laptop, not every saved credential to every box you own.

> ### *The Essential Eight Angle in One Sentence*
>
> *The Essential Eight's restrict administrative privileges strategy is much easier to evidence when administration only ever happens from one hardened, logged, tightly controlled system. A jump host turns a policy statement into a network fact.*

The Ubuntu 26.04 Problem: Wayland

Ubuntu 26.04 ships Wayland-native. There is no X session to speak of in the default install, and that matters because xrdp is an X11 technology at heart.

The built-in GNOME remote desktop server (gnome-remote-desktop) does speak RDP and works over Wayland, but it is oriented towards a single user: it is designed around mirroring or extending one person's console session, not hosting half a dozen concurrent logins. For a multi-user jump host it is the wrong tool.

The good news is that xrdp plus the xorgxrdp driver does not care what your console session runs. Each incoming RDP connection gets its own dedicated Xorg server, entirely separate from the physical display. Wayland can have the console; xrdp will run its own X11 sessions in parallel for as many users as you throw at it. We just need a desktop environment that still speaks X11, which leads to the next decision.

Choosing a Desktop: XFCE, and What Else I Looked At

The desktop on a jump host has an unusual job description: it needs to be light (RAM is shared between concurrent sessions), it needs to be X11-native (no Wayland-only surprises), and it needs to be boring enough that anyone landing on it can find a terminal and a browser without a training course.

- **XFCE** is the sweet spot. X11-native, mature over xrdp, modest memory footprint per session, and a traditional panel-and-menu layout that nobody needs help with. This is what I went with.
- **LXQt** is lighter again, but Qt theming over RDP has its quirks and the ecosystem is smaller.
- **MATE** is a fine X11 desktop and works well with xrdp, but it is heavier than XFCE for no real gain in this role.
- **IceWM or Openbox** are ultra-light and honestly tempting for a box whose job is mostly launching terminals, but they are austere enough that I would spend more time explaining the desktop than using it.
- **GNOME and KDE Plasma** over xrdp are possible with varying degrees of pain, but both are heavy per session and increasingly Wayland-first, which is exactly what we are trying to sidestep.

For a multi-user host where every session costs RAM, XFCE wins on the balance of weight, compatibility and familiarity.

The Build

Install xrdp, the Xorg backend, and the desktop:

```
sudo apt install xrdp xorgxrdp xfce4 xfce4-goodies
sudo systemctl enable --now xrdp
```

The stock `/etc/xrdp/startwm.sh` tries to be clever about detecting a desktop session, and on 26.04 that cleverness works against you. Replace it with something explicit:

```
#!/bin/sh
if [ -r /etc/profile ]; then
 . /etc/profile
fi
# xrdp Xorg authentication
export XAUTHORITY="$HOME/.Xauthority"
# XFCE environment
export XDG_CURRENT_DESKTOP=XFCE
export XDG_SESSION_DESKTOP=xfce
export DESKTOP_SESSION=xfce
# Pass the RDP environment to the user's systemd/dbus session
dbus-update-activation-environment --systemd DISPLAY XAUTHORITY XDG_CURRENT_DESKTOP
XDG_SESSION_DESKTOP DESKTOP_SESSION
exec startxfce4
```

A few things worth explaining in that script. The `XAUTHORITY` export avoids a class of confusing authentication failures where the session goes looking for the X magic cookie in the wrong place. The three desktop environment variables stop applications from misdetecting the session and falling back to generic behaviour. And the `dbus-update-activation-environment` line hands the RDP session's environment across to the user's systemd and D-Bus session, which matters more than it used to: without it, snap-aware and portal-aware applications can launch into a void.

Two more bits of housekeeping. The xrdp service needs to read its TLS key, which on Ubuntu lives in the ssl-cert group:

```
sudo adduser xrdp ssl-cert
sudo ufw allow from 10.10.10.0/24 to any port 3389 proto tcp
sudo systemctl restart xrdp
```

Note the firewall rule restricts 3389 to the management subnet rather than opening it to the world. This box is supposed to reduce your attack surface, not add to it. (In part 2, Guacamole will be the only thing that talks to 3389 at all, and it will live on the same host.)

One known annoyance: fresh xrdp sessions on Ubuntu love to pop a polkit prompt about "creating a colour managed device" on login. It is harmless and fixable with a small polkit rule for the colord actions; I will cover that in part 2 when I tidy the box up for real users.

Alternatives Worth Knowing About

xrdp plus XFCE is not the only way to solve this, and in the spirit of due diligence I looked at three alternatives before settling.

Kasm Workspaces takes a different architectural approach: instead of one multi-user box, every user session is a fresh, disposable Docker container streaming a desktop (or just a single app) to the browser. The isolation story is genuinely better - a compromised session is a throwaway container, not a foothold on your shared jump host - and browser delivery is built in, so you would not need Guacamole. The trade-offs are resource overhead (a container per session adds up fast), more moving parts to operate, and the fact that the most interesting features sit in the paid tiers. If I were building this for a larger team, Kasm would be on the shortlist. For a handful of admins it is more platform than I need.

Teleport is a different category again. It is an identity-aware access proxy: short-lived certificates, RBAC, session recording and a proper audit trail across SSH, Kubernetes, databases and Windows desktops over RDP. If your driver for a jump host is audit and compliance, Teleport attacks the same problem from the identity side and does it well. Two caveats stopped me. First, its graphical desktop support is Windows-focused; there is no Linux desktop equivalent, so it would not replace this box for GUI work. Second, the licensing of the community binaries has some sharp edges worth reading before you build on it. Teleport is where I would go to govern SSH access at scale; it does not give you a shared Linux desktop to land on.

X2Go deserves an honourable mention as the other serious multi-user Linux remote desktop protocol. It is NX-based, performant over slow links, and handles concurrent sessions well. What killed it for me is the client story: it needs the X2Go client installed on every machine, whereas every Windows box already ships an RDP client, and my end goal is browser access via Guacamole, which speaks RDP natively.

None of these change my answer for this build, but they define the upgrade paths: Kasm if I outgrow a shared box, Teleport if audit becomes the primary requirement, X2Go if I ever control the client fleet.

Where Things Stand

The jump host now accepts multiple concurrent RDP sessions, each user landing in their own XFCE desktop on their own Xorg server, with the Wayland console none the wiser. The next step is making it reachable without an RDP client at all: Apache Guacamole, served over HTTPS, rendering these same sessions in a browser tab, with per-user authentication and connection logging in one place. That is part 2.

Related:

- ACSC Guidelines for System Management - the ISM guidance behind segregating administrative environments
- Kasm Workspaces - containerised desktop streaming, the disposable-session alternative
- Teleport - identity-aware access proxy for SSH, Kubernetes and Windows RDP