Remote Desktop: RDP & VNC

A Complete Practical Guide

8 Chapters

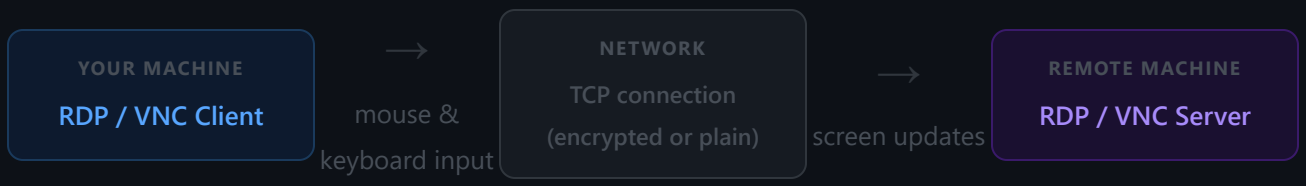
Chapter 1 — Remote Desktop Concepts

SSH gives you a terminal — text in, text out. Remote desktop goes further: it lets you see and interact with a full graphical desktop on a remote machine as if you were sitting in front of it. Mouse movements, window clicks, keyboard input — all transmitted across the network in real time.

Two protocols dominate the field: **RDP** (Remote Desktop Protocol, developed by Microsoft) and **VNC** (Virtual Network Computing, an open standard). They solve the same problem in quite different ways, with different performance characteristics, security models, and ideal use cases.

How Remote Desktop Works

At its core, every remote desktop protocol follows the same pattern: the server captures what's on screen, compresses and encodes it, and sends it to the client. The client displays it and sends back user input (mouse, keyboard). The differences between protocols lie in *how* they do each step.



RDP vs VNC — The Two Protocols

RDP

Remote Desktop Protocol · Microsoft · Port 3389

Developed by Microsoft (1998)

VNC

Virtual Network Computing · Open standard · Port 5900

How it works	Sends rendering instructions (draw this rectangle, render this font) — not raw pixel data
Performance	Excellent — very efficient on slow connections
Encryption	Built-in TLS — encrypted by default
Platforms	Native on Windows; clients available everywhere; server needs xRDP on Linux
Sessions	Creates a separate session — doesn't show what's on the physical screen

Developed by	Olivetti Research Lab (1998), now open standard
How it works	Sends raw pixel data (screenshots of changed regions) — framebuffer sharing
Performance	Heavier — pixel data is large; better on fast local networks
Encryption	NOT encrypted by default — must tunnel through SSH
Platforms	Truly cross-platform — Linux, macOS, Windows, Raspberry Pi
Sessions	Mirrors the physical screen — you see exactly what's on the monitor

> **The key architectural difference:** RDP sends drawing commands ("render a button here, draw this text there") and the client reconstructs the UI locally. VNC sends raw pixel snapshots of what changed on screen. RDP is more like a shared app; VNC is more like a video feed of the screen. This is why RDP is faster over the internet and VNC is better for seeing exactly what's physically on the monitor.

Side-by-Side Comparison

Feature	RDP	VNC
Default port	3389	5900 (5901, 5902… per display)
Encryption	Built-in TLS	None by default — use SSH tunnel
Performance	Excellent on slow links	Heavier — needs fast connection
Linux server	Needs xRDP installed	Native (TigerVNC, RealVNC etc.)
Windows server	Built in (Pro/Server editions)	Third-party software needed

Feature	RDP	VNC
Screen mirroring	Separate session only	Mirrors physical display
Multi-monitor	Supported natively	Limited / implementation-dependent
Audio redirection	Supported	Limited
File transfer	Built in (drive redirection)	Not built in — use SFTP separately
Cross-platform clients	Good — many available	Excellent — universal standard
Raspberry Pi / headless Linux	Possible but extra setup	Ideal use case

Use Cases — Which Protocol Fits

💼

Accessing your work Windows PC from home

Full desktop access to a PC that stays at the office, with file access, audio, clipboard sync.

RDP — BUILT IN, FAST, ENCRYPTED

🍓

GUI access to a Raspberry Pi

Your Pi is headless (no monitor). You want to see its desktop without plugging in a screen.

VNC — MIRRORS THE DISPLAY, WORKS GREAT ON LAN

🖥️

Helping a family member with their PC

You need to see exactly what's on their screen — the same view they have.

VNC — SCREEN MIRROR. OR TEAMVIEWER/ANYDESK FOR SIMPLICITY

🐧

Linux desktop from Windows

Accessing a Debian or Ubuntu desktop remotely from a Windows machine.

EITHER — XRDP FOR PERFORMANCE, VNC FOR SIMPLICITY

🔧

Remote server administration with a GUI

Running a GUI tool (database manager, file manager) on a headless server.

RDP VIA XRDP — CLEANER SESSIONS, BETTER PERFORMANCE

🌐

Access over the internet

Connecting to a machine from outside your home network — over a VPN or SSH tunnel.

RDP PREFERRED — LIGHTER BANDWIDTH, BUILT-IN ENCRYPTION

Security Considerations

Remote desktop is one of the most attacked surfaces on the internet. A Windows machine with RDP exposed on port 3389 will be hit by automated brute-force bots within minutes of going online. VNC without a tunnel is even worse — it has no built-in encryption and weak default authentication. These protocols were designed for local networks first.

CRITICAL

Exposing RDP directly to the internet

Port 3389 open on your firewall invites constant automated brute-force attacks. BlueKeep (CVE-2019-0708) and DejaBlue are examples of critical RDP vulnerabilities. Never expose it publicly without a VPN or reverse tunnel.

CRITICAL

VNC without SSH tunnel

Plain VNC sends pixel data unencrypted. Passwords are weakly hashed. Anyone on the same network can intercept the session. Always tunnel VNC through SSH — covered in Chapter 6.

HIGH

Weak passwords

Both RDP and VNC are targeted by credential-stuffing attacks. Use strong, unique passwords and — where possible — restrict access to known IPs via firewall rules.

HIGH

Unpatched servers

RDP has a history of critical remote-code-execution vulnerabilities. Keep Windows and xRDP updated. Enable Windows Update and don't defer security patches on machines with RDP enabled.

MEDIUM

No MFA on RDP

Windows RDP supports Network Level Authentication (NLA) which requires credentials before the session starts — harder to exploit. Enable NLA and consider adding MFA via a VPN layer.

MEDIUM

Leaving sessions open

An unlocked remote desktop session is a wide-open door. Configure auto-lock on inactivity and disconnect sessions when not in use. On Linux, VNC servers should be stopped when not needed.

> **The golden rule:** Never expose RDP (port 3389) or VNC (port 5900) directly to the internet. Always access them through a VPN, an SSH tunnel, or a reverse proxy with strong authentication in front. This course covers both approaches in later chapters.

What This Course Covers

Eight chapters covering both protocols across the platforms you're most likely to encounter:

- **Chapters 2–4** — RDP in depth: enabling it on Windows, connecting from Linux and from Windows, multi-monitor and audio settings

- **Chapter 5** — xRDP: bringing RDP to Linux (Debian/Ubuntu)
- **Chapter 6** — VNC: TigerVNC setup and the essential SSH tunnel
- **Chapter 7** — Alternatives: NoMachine, AnyDesk, TeamViewer — when the classics aren't the right tool
- **Chapter 8** — Getting through firewalls: SSH tunnels, Cloudflare Tunnel, performance tuning on slow connections

Next — Chapter 2: Windows RDP. RDP has been built into Windows since XP, but it's hidden behind a few settings and only available on Pro and Server editions. Chapter 2 covers enabling Remote Desktop, the Network Level Authentication setting, firewall rules, connecting with `mstsc`, and the most useful connection options — display resolution, local drive access, and clipboard sharing.

Chapter 2 — Windows RDP

Remote Desktop Protocol is built into every modern version of Windows — but it's disabled by default and locked to specific editions. This chapter covers enabling it, configuring it securely with Network Level Authentication, connecting with the `mstsc` client, and the most useful options: display scaling, local drive access, clipboard sharing, and saved connection files.

Which Windows Editions Support RDP

RDP has two sides: the **client** (connecting to another machine) and the **host** (being connected to). The client is available on all editions. The host — actually accepting incoming connections — is restricted to Pro and Server editions.

Windows 11 Pro	Windows 10 Pro	Windows Server	Windows 11 Home	Windows 10 Home
Host + Client ✓	Host + Client ✓	Host + Client ✓	Client only ✗	Client only ✗

> **Home edition workaround:** If your target machine runs Windows Home, you can still get RDP-like access using **RDP Wrapper** (a third-party tool that unlocks the hidden RDP service on Home editions) or switch to VNC instead. Chapter 6 covers VNC.

Enabling Remote Desktop on the Host Machine

1. **Open System Properties — Remote tab**

   Settings → System → Remote Desktop (Windows 11)
   or: right-click This PC → Properties → Remote Settings

Toggle **Enable Remote Desktop** to On. Windows will warn you that the PC must be discoverable on the network — confirm.

2 **Keep Network Level Authentication enabled**

Same panel — "Require devices to use Network Level Authentication to connect"

Leave this **checked**. NLA requires credentials before the remote session even starts — it significantly reduces the attack surface. Only disable it if you're connecting from a very old client that doesn't support it.

3 **Add authorised users (if not connecting as admin)**

Remote Desktop settings → "Select users that can remotely access this PC"

Administrators can always connect. For other accounts, click **Add** and type the username. On a domain machine, use `DOMAIN\username`.

4 **Check Windows Firewall allows RDP**

Windows Defender Firewall → Allow an app → Remote Desktop

Enabling Remote Desktop in Settings usually does this automatically. Verify the **Remote Desktop** rule is ticked for the appropriate network profile (Private for home, Domain for work). Leave Public **off** unless you have a specific reason.

5 **Note the machine name or IP address**

Settings → System → About → Device name (or run ipconfig)

You'll need either the machine's hostname (e.g. `PHILIPS-PC`) or its IP address (`192.168.0.50`) to connect. On a local network, the hostname usually works. Over the internet, use the IP or a dynamic DNS name.

> **The machine must stay awake** to accept connections. Go to Power Settings and set **Sleep → Never** (or at least a long timeout) on the host machine. A sleeping PC drops all RDP connections and can't be woken remotely unless Wake-on-LAN is configured.

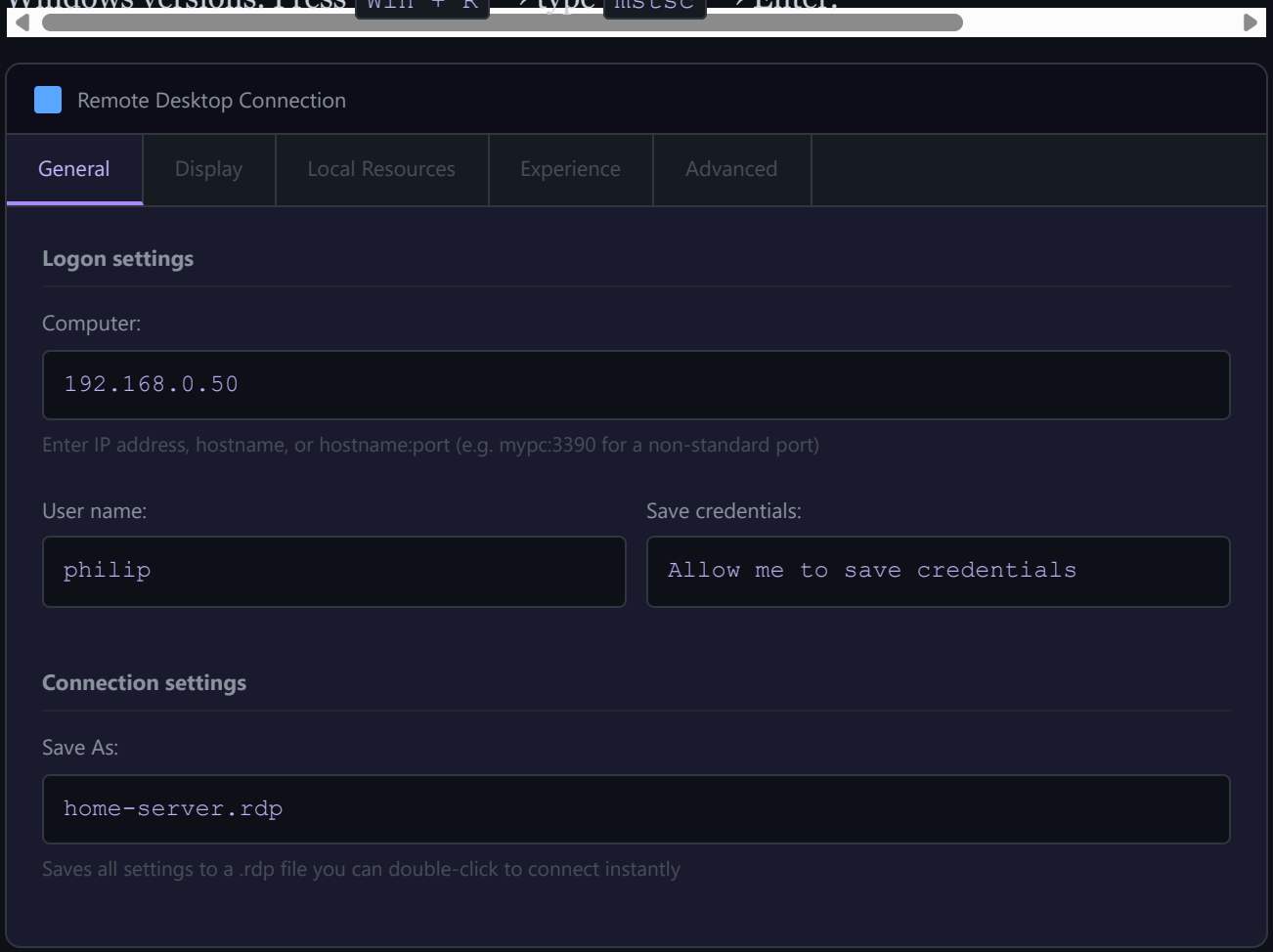
Enabling via PowerShell (Headless / Scripted)

```
# Enable Remote Desktop
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server'
  -Name "fDenyTSConnections" -Value 0

# Enable NLA (recommended)
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server\W
```

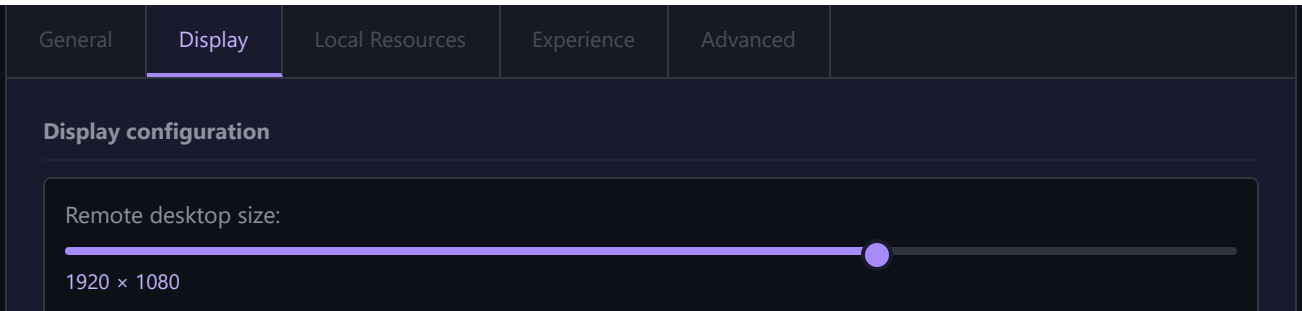
```
  -Name "UserAuthentication" -Value 1

# Allow RDP through Windows Firewall
Enable-NetFirewallRule -DisplayGroup "Remote Desktop"

# Verify RDP is enabled
Get-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server'
  -Name "fDenyTSConnections"
# fDenyTSConnections = 0 means RDP is enabled
```

Connecting with mstsc

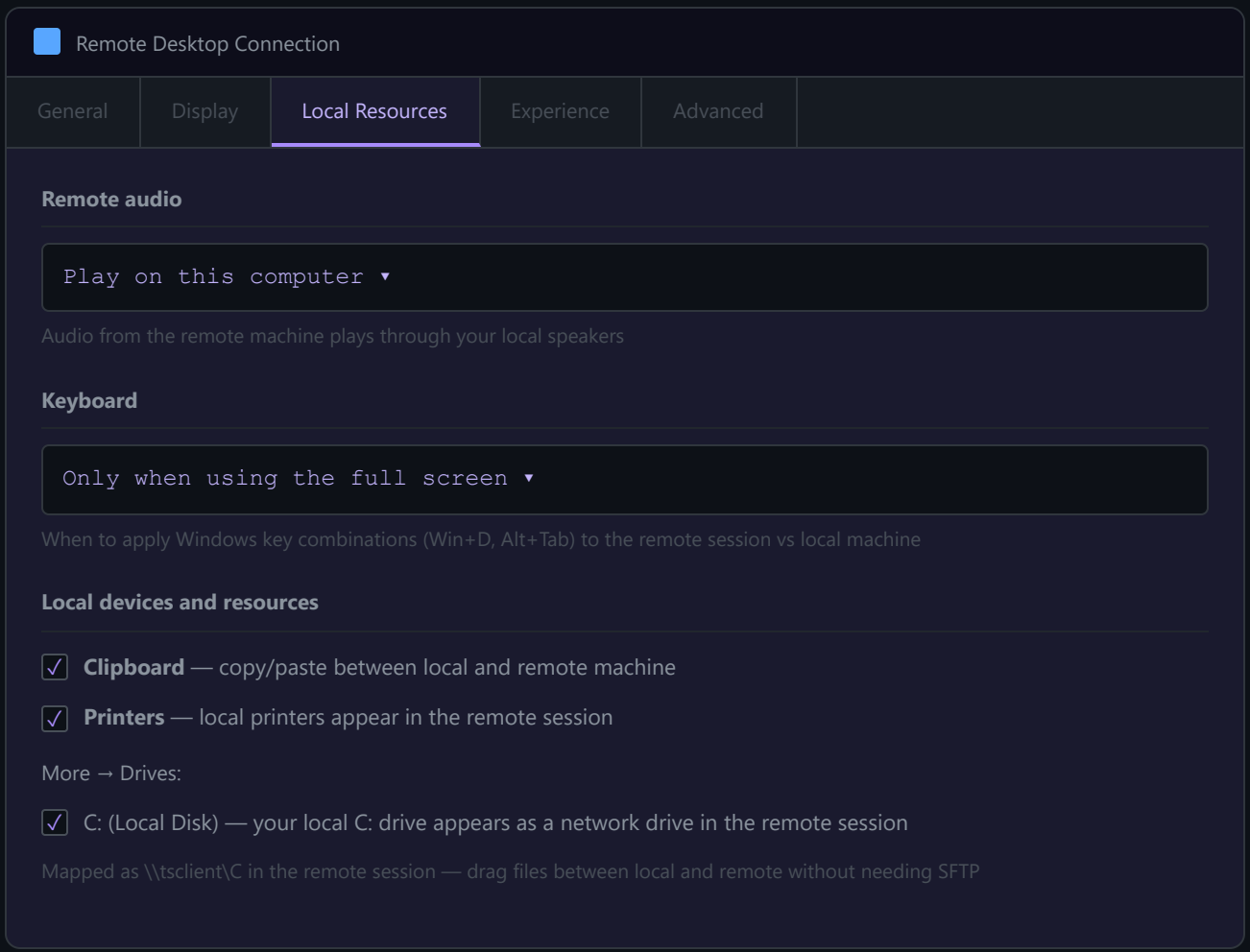
`mstsc.exe` — Microsoft Terminal Services Client — is the built-in RDP client on all Windows versions. Press `Win + R` → type `mstsc` → Enter.



Display tab — resolution and colour



Local Resources tab — clipboard, drives, audio



Saving Connections as .rdp Files

Every setting in `mstsc` can be saved to a `.rdp` file. Double-click the file to launch the connection with all settings pre-loaded — no re-entering hostnames or tweaking options each time.

```
# A .rdp file is plain text — you can edit it directly
# home-server.rdp

full address:s:192.168.0.50
username:s:philip
desktopwidth:i:1920
desktopheight:i:1080
session bpp:i:32
audiomode:i:0           # 0=play locally, 1=play on server, 2=no audio
redirectclipboard:i:1    # 1=enable clipboard sharing
redirectdrives:i:1       # 1=share local drives
drivestoredirect:s:C:\; # which drives to share
use multimon:i:1         # 1=use all monitors
screen mode id:i:2       # 1=windowed, 2=full screen
networkautodetect:i:1    # auto-tune for network speed
```

> **Keep a .rdp file per machine.** Put them in a folder (`~/RDP Connections/`), pin the folder to Quick Access, and connecting to any machine becomes a single double-click. Name them clearly: `home-server.rdp`, `work-laptop.rdp`, `pi-desktop.rdp`.

mstsc Command-Line Options

Command	What it does
`mstsc`	Open the connection dialog
`mstsc /v:hostname`	Connect directly to a host
`mstsc /v:host:3390`	Connect on a non-standard port
`mstsc /f`	Full screen mode
`mstsc /w:1280 /h:720`	Set window width and height

Command	What it does
`mstsc /multimon`	Use all local monitors
`mstsc home-server.rdp`	Open a saved .rdp file
`mstsc /edit home-server.rdp`	Open .rdp file in the settings dialog rather than connecting
`mstsc /admin`	Connect to admin session (bypasses session limits on Server editions)
`mstsc /restrictedAdmin`	Restricted admin mode — credentials not sent to remote host (pass-the-hash protection)

Useful Keyboard Shortcuts Inside an RDP Session

Shortcut	What it does (inside RDP session)
`Ctrl + Alt + Break`	Toggle between full screen and windowed mode
`Ctrl + Alt + Home`	Activate the connection bar (when in full screen)
`Alt + Page Up / Down`	Switch between apps on the remote machine (replaces Alt+Tab)
`Alt + Delete`	Open the remote machine's window menu (replaces Alt+F4)
`Ctrl + Alt + End`	Send Ctrl+Alt+Del to the remote machine
`Ctrl + Alt + Minus (numpad)`	Take a screenshot of the active remote window
`Ctrl + Alt + Plus (numpad)`	Take a screenshot of the entire remote desktop

Security — Tightening Up Before You Connect

- **Keep NLA enabled.** Network Level Authentication requires credentials before the session opens — attackers can't even get to the login screen without valid credentials.
- **Use a strong password** on the account being used for RDP. Brute-force is the most common attack vector.
- **Don't use port 3389 publicly.** On a home network this is fine. Over the internet, use a VPN or SSH tunnel instead of opening port 3389 on your router. Chapter 8 covers this.
- **Limit which users can connect.** Don't leave "All users" — explicitly add only the accounts that need remote access.

- **Check Windows Update** before enabling RDP. RDP has a history of critical vulnerabilities; being fully patched is non-negotiable.

> **Never forward port 3389 on your home router to a Windows machine.** Bots scan for open RDP ports constantly. Within hours of going live, an exposed RDP machine will be under sustained brute-force attack. If you need RDP over the internet, always put it behind a VPN or an SSH tunnel first.

Next — Chapter 3: Connecting to Windows from Linux. The `mstsc` client only runs on Windows. From a Linux machine (or your Debian server), you connect to Windows RDP using **Remmina** — a full-featured GUI client — or **xfreerdp** on the command line. Chapter 3 covers both, including audio redirection, drive sharing, and connecting through non-standard ports.

Chapter 3 — Connecting to Windows from Linux

The `mstsc` client is Windows-only. From a Linux machine — your Debian server, a Raspberry Pi, or a Linux laptop — you connect to a Windows RDP host using either **Remmina** (a polished GUI client) or **xfreerdp** (the powerful command-line tool that Remmina itself uses under the hood). Both are free, open source, and available in every major distro's package manager.

Remmina vs xfreerdp — Which to Use

Remmina

GUI CLIENT · GTK

Saved connection profiles, drag-and-drop file transfer, tabbed sessions, system tray integration. Supports RDP, VNC, SSH, SPICE — one app for all remote access needs.

✓ Best for daily interactive use
✓ Stores credentials securely
⚠ Needs a desktop environment (not headless)

xfreerdp

COMMAND-LINE · FREERDP

Full RDP implementation with fine-grained flag control. Scriptable, works over SSH, usable from a headless server. Remmina calls xfreerdp internally for its RDP connections.

✓ Scriptable and automatable
✓ Works without a desktop environment
⚠ No session management or saved profiles

Installing Both

philip@debian — installing Remmina and xfreerdp

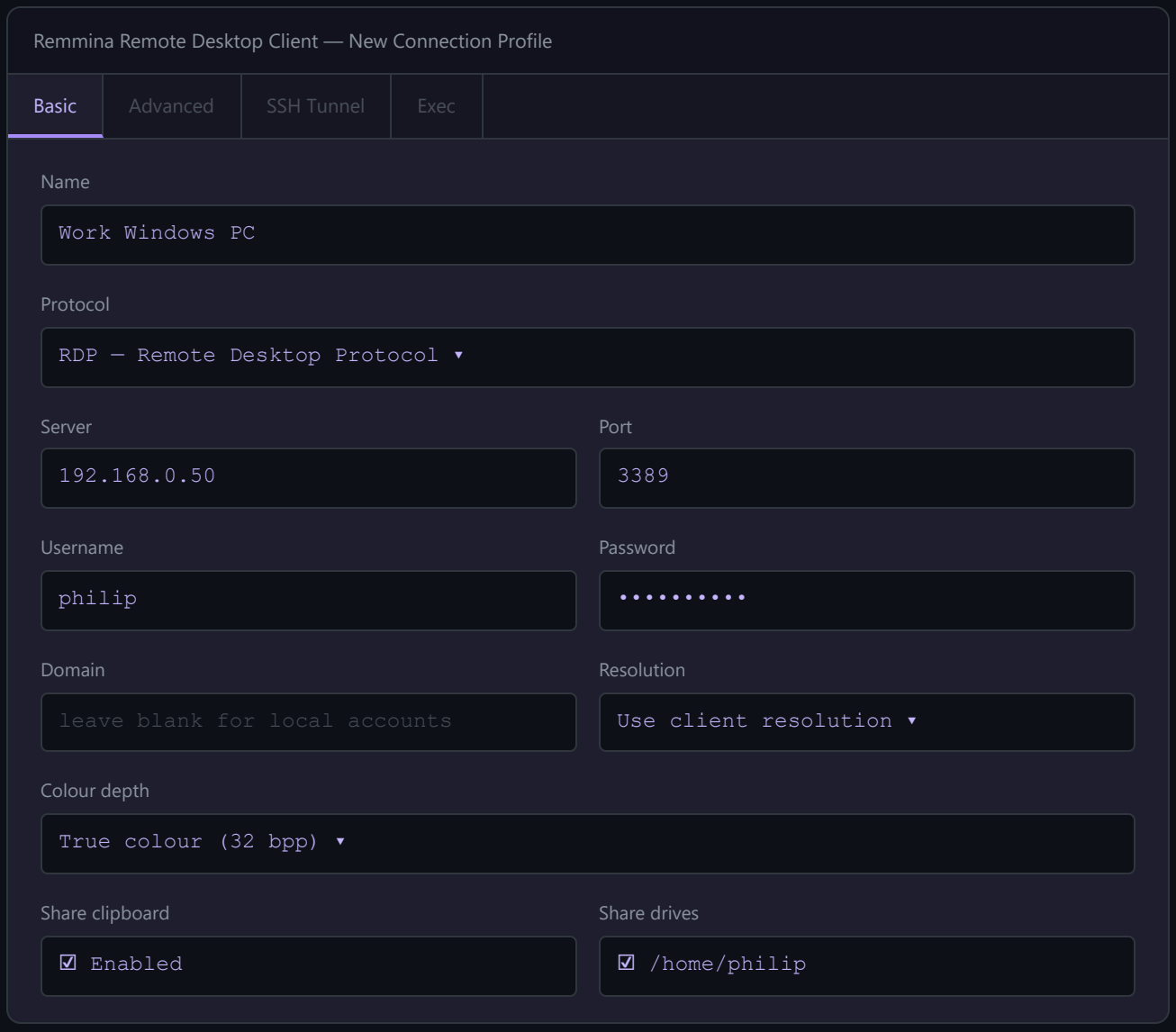
```
# Debian / Ubuntu philip@debian:~$ sudo apt update && sudo apt install remmina remmina-plugin-rdp freerdp2-x11 # Fedora / RHEL philip@fedora:~$ sudo dnf install remmina remmina-plugins-rdp freerdp # Arch philip@arch:~$ sudo pacman -S remmina freerdp # Confirm xfreerdp is available philip@debian:~$ xfreerdp --version This is FreeRDP version 2.11.2 (git n/a)
```

FreeRDP vs freerdp2: Debian/Ubuntu package the command as `xfreerdp` from the `freerdp2-x11` package. On newer systems you may find `xfreerdp3` from FreeRDP 3.x. The

> flags are nearly identical — the examples in this chapter work on both versions.

Using Remmina

Launch Remmina from your applications menu or run `remmina` in a terminal. Click the + button to create a new connection profile:



Save the profile and double-click it to connect. Remmina stores all profiles in `~/.local/share/remmina/` — back them up or sync them between machines to carry your connection list with you.

The SSH Tunnel tab

Remmina has built-in SSH tunnelling — go to the **SSH Tunnel** tab in the connection profile and tick **Enable SSH tunnel**. Set the SSH server details and Remmina will automatically open the tunnel before connecting RDP through it. This is the cleanest way to use RDP securely over the internet without any manual port-forwarding setup.

Using xfreerdp on the Command Line

xfreerdp is the direct CLI tool. The basic connection is straightforward; additional flags layer on features:

philip@debian — xfreerdp examples

```
# Minimal connection — will prompt for password philip@debian:~$ xfreerdp
/v:192.168.0.50 /u:philip # Full desktop, clipboard, drives, audio — the everyday
command philip@debian:~$ xfreerdp /v:192.168.0.50 /u:philip /p:'MyPassword' \
/w:1920 /h:1080 \ +clipboard \ /drive:home,/home/philip \ /sound:sys:alsa # Full
screen philip@debian:~$ xfreerdp /v:192.168.0.50 /u:philip /f # Non-standard port
philip@debian:~$ xfreerdp /v:192.168.0.50:3390 /u:philip # Dynamic resolution —
window resizes update remote desktop size philip@debian:~$ xfreerdp
/v:192.168.0.50 /u:philip /dynamic-resolution # Ignore certificate warning (self-
signed or NLA mismatch) philip@debian:~$ xfreerdp /v:192.168.0.50 /u:philip
/cert:ignore
```

Sharing local folders with the remote Windows machine

philip@debian — sharing drives via xfreerdp

```
# Share a single directory — appears as a network drive in Windows Explorer
philip@debian:~$ xfreerdp /v:192.168.0.50 /u:philip \
/drive:LinuxHome,/home/philip # In Windows: This PC → LinuxHome
(\\tsclient\LinuxHome) # Share multiple directories philip@debian:~$ xfreerdp
/v:192.168.0.50 /u:philip \ /drive:Projects,/home/philip/projects \
/drive:Downloads,/home/philip/Downloads # Share entire filesystem (useful for
file transfers) philip@debian:~$ xfreerdp /v:192.168.0.50 /u:philip \
/drive:Linux,/
```

xfreerdp Flag Reference

Flag	What it does	Example
`/v:host`	Target host — IP, hostname, or hostname:port	`/v:192.168.0.50`
`/u:user`	Username on the Windows machine	`/u:philip`
`/p:pass`	Password (omit to be prompted securely)	`/p:'MyPass!'`
`/d:domain`	Domain — leave out for local accounts	`/d:CORP`
`/w: /h:`	Window width and height in pixels	`/w:1920 /h:1080`
`/f`	Full screen mode	`/f`
`/dynamic-resolution`	Remote desktop resizes when you resize the window	`/dynamic-resolution`
`/bpp:16`	Colour depth — 16 reduces bandwidth on slow links	`/bpp:16`
`+clipboard`	Enable clipboard sharing (copy/paste both ways)	`+clipboard`
`/drive:name,path`	Share a local Linux directory as a Windows network drive	`/drive:home,/home/philip`
`/sound:sys:alsa`	Redirect remote audio to local ALSA output	`/sound:sys:alsa`
`/sound:sys:pulse`	Redirect remote audio to local PulseAudio	`/sound:sys:pulse`
`/microphone`	Redirect local microphone to the remote machine	`/microphone`
`/printer`	Redirect local printers to remote session	`/printer`
`/cert:ignore`	Skip certificate verification (use on trusted networks only)	`/cert:ignore`
`/cert:tofu`	Trust on first use — accept and remember the certificate	`/cert:tofu`
`/compression`	Enable RDP compression — useful on slow connections	`/compression`
`-wallpaper`	Disable remote desktop wallpaper (speeds up redraws)	`-wallpaper`
`-themes`	Disable visual themes on remote (further speeds redraws)	`-themes`

> **Prefix rules:** xfreerdp uses `/flag` for most options, `+flag` to explicitly enable something, and `-flag` to explicitly disable it. So `+clipboard` enables clipboard, `-wallpaper` disables the desktop wallpaper.

Saving a Command as a Shell Alias

xfreerdp has no built-in profile system, but a shell alias or script gives you the same single-command convenience:

```
# Add to ~/.bashrc or ~/.bash_aliases
alias rdp-work='xfreerdp /v:192.168.0.50 /u:philip +clipboard /dynamic-resolutic
alias rdp-server='xfreerdp /v:192.168.0.24 /u:philip +clipboard /w:1600 /h:900'

# Reload and use
source ~/.bashrc
rdp-work      # connects and prompts for password
```

Performance Tuning on Slow Connections

```
# Lean profile for slow/remote connections
xfreerdp /v:host /u:user \
  /bpp:16 \           # 16-bit colour — half the pixel data
  /compression \      # compress the data stream
  -wallpaper \        # no desktop wallpaper
  -themes \           # flat window decorations
  -aero \             # no Windows Aero effects
  -fonts \            # no ClearType (font smoothing)
  -animations         # no UI animations
```

Common Errors and Fixes

ERRCONNECT_LOGON_FAILURE

Wrong username or password

Double-check credentials. If using a Microsoft account, try the local account name only (not the email). On a domain machine use `/d:DOMAIN /u:user`.

Certificate / TLS error on connection

Self-signed cert or NLA mismatch

Add `/cert:tofu` to accept and remember the certificate. On a trusted LAN, `/cert:ignore` skips the check entirely.

ERRCONNECT_CONNECT_FAILED

Can't reach the host — network or firewall

CredSSP / NLA auth failure

Windows patched, CredSSP version mismatch

Confirm the host is on and reachable: `ping 192.168.0.50`. Check Windows Firewall has the Remote Desktop rule enabled for the correct network profile.

Add `/sec:rdp` to fall back to standard RDP security instead of NLA. Not ideal for production — update FreeRDP if possible.

Black screen after connecting

Display driver or GPU acceleration issue

Add `/gdi:sw` to force software rendering instead of hardware-accelerated GDI. Slower but reliable.

No audio / sound not redirected

Wrong audio system specified

Try both `/sound:sys:alsa` and `/sound:sys:pulse`. Run `pactl info` to confirm whether PulseAudio or PipeWire is your audio server.

Quick Reference

Task	Command
Install on Debian/Ubuntu	`sudo apt install remmina remmina-plugin-rdp freerdp2-x11`
Basic xfreerdp connection	`xfreerdp /v:192.168.0.50 /u:philip`
Full-featured connection	`xfreerdp /v:host /u:user +clipboard /dynamic-resolution /sound:sys:pulse`
Share a local folder	`xfreerdp /v:host /u:user /drive:name,/local/path`
Non-standard port	`xfreerdp /v:host:3390 /u:user`
Skip certificate check	`xfreerdp /v:host /u:user /cert:ignore`
Slow-connection profile	`xfreerdp /v:host /u:user /bpp:16 /compression -wallpaper -themes`
Save as alias	`alias rdp-home='xfreerdp /v:host /u:user +clipboard'`

Next — Chapter 4: Connecting to Windows from Windows. `mstsc` in depth — saved .rdp files, multi-monitor spanning, audio redirection, RemoteApp for running individual Windows apps rather than a full desktop, and the Remote Desktop app from the Microsoft Store (the modern replacement for mstsc).

Chapter 4 — Connecting to Windows from Windows

Chapter 2 covered enabling RDP on the host machine. This chapter goes deeper into the client side — getting the most out of `mstsc` (the built-in client), the Experience tab's performance presets, spanning multiple monitors, audio and device redirection, and RemoteApp for running individual Windows applications as if they were local. We also look at the modern **Microsoft Remote Desktop** app from the Store, which replaces mstsc for some workflows.

The Experience Tab — Performance Presets

The Experience tab in `mstsc` controls which visual effects are enabled during the session. On a fast LAN, keep everything on. Over a slower connection, disabling effects dramatically reduces bandwidth and makes the session feel more responsive.

LAN (10 MBPS OR HIGHER)

Everything enabled

Desktop background, font smoothing, desktop composition, window animations, themes, bitmap caching. Full visual experience — looks identical to sitting at the machine.

BROADBAND (2–10 MBPS)

Themes + font smoothing only

Desktop composition and animations disabled. Looks slightly flatter but remains comfortable. Good preset for home broadband or a business VPN connection.

MODEM / SLOW LINK (< 2 MBPS)

Bitmap cache only

Everything visual is stripped out. Only bitmap caching (storing frequently-used screen regions locally) remains. Functional but spartan. For mobile data or a slow VPN.

CUSTOM

Pick and mix

Manually tick each option. Recommended: keep **bitmap caching** always on — it caches repeated UI elements and significantly reduces redraws at no perceptible quality cost.

> **The single best setting for slow connections:** disable the desktop background (wallpaper) and font smoothing — these two alone account for a large fraction of bandwidth. Keep themes and bitmap caching. The session will feel much snappier without looking completely bare.

Multi-Monitor Support

mstsc can span a remote desktop session across all your local monitors, making the remote machine feel fully integrated with your workspace:


```
# Use all monitors from the command line
mstsc /multimon /v:hostname

# Or in a .rdp file
use multimon:i:1
screen mode id:i:2    # 2 = full screen

# Span specific monitors only (by monitor ID)
# Use mstsc /l to list monitor IDs on your machine
mstsc /l
```

> **Monitor alignment matters.** Windows requires all monitors in a multi-monitor RDP session to be the same resolution, or the session may only span correctly on matched displays. Mismatched resolutions can cause the remote desktop to appear on only one monitor or scale oddly. If this happens, switch to a single-monitor full-screen session and adjust display settings on the remote machine first.

Selective monitor spanning — choose which monitors to use

```
# selectedmonitors — comma-separated list of monitor IDs from mstsc /l
selectedmonitors:s:0,1    # use monitor 0 and 1 only, skip monitor 2
use multimon:i:1
```

Audio and Device Redirection in Depth

Audio — three modes

```
# In a .rdp file — audiomode controls where remote audio plays
audiomode:i:0     # Play on this computer (default) — audio comes out your local
audiomode:i:1     # Play on remote computer — audio stays on the server's speaker
audiomode:i:2     # Do not play — no audio at all

# Microphone redirection — send your local mic to the remote session
audiocapturemode:i:1     # 1 = redirect microphone, 0 = no mic
```

Drive redirection

```
# In a .rdp file
redirectdrives:i:1              # enable drive redirection
drivestoredirect:s:C:\;         # share local C: drive only
drivestoredirect:s:*             # share ALL local drives
drivestoredirect:s:DynamicDisks # share USB drives as they are plugged in

# On the remote machine, shared drives appear in File Explorer under:
# This PC → philip on LAPTOP-NAME (\\tsclient\C)
```

Other device redirection

```
# Clipboard — almost always want this on
redirectclipboard:i:1

# Printers — local printers appear in the remote session
redirectprinters:i:1

# Smart cards (for corporate authentication)
redirectsmartcards:i:1

# USB devices (requires RD Gateway or RemoteFX — not always available)
usbdevicestoredirect:s:*

# Webcam redirection (Windows 10 v1803+ with server support)
camerastoredirect:s:*
```

The Complete .rdp File Reference

A `.rdp` file is plain text — every setting mstsc can save is a key:type:value line. Here is a complete working file for a power-user setup:

```
# work-pc.rdp — full-featured connection profile

## Connection
full address:s:192.168.0.50
username:s:philip
domain:s:                          # blank for local account
prompt for credentials:i:0         # 0=use saved creds, 1=always prompt

## Display
screen mode id:i:2                 # 1=windowed, 2=fullscreen
use multimon:i:1                   # span all monitors
desktopwidth:i:1920
desktopheight:i:1080
session bpp:i:32                   # colour depth
smart sizing:i:0                   # 1=scale remote to fit window
dynamic resolution:i:1             # resize remote when window resizes

## Performance
connection type:i:7                # 1=modem … 7=LAN, auto-tunes experience
networkautodetect:i:1              # auto-detect bandwidth
bandwidthautodetect:i:1
bitmapcachepersistenable:i:1       # persist bitmap cache between sessions

## Audio / Video
audiomode:i:0                      # play audio locally
audiocapturemode:i:1               # redirect microphone
videoplaybackmode:i:1              # optimise video playback

## Devices
redirectclipboard:i:1
redirectdrives:i:1
drivestoredirect:s:C:\;
redirectprinters:i:1
redirectsmartcards:i:0
camerastoredirect:s:*

## Security
authentication level:i:2           # 0=no auth, 1=warn, 2=require NLA
```

```
enablecredsspsupport:i:1           # use CredSSP (NLA)
disableconnectionsharing:i:0
```

RemoteApp — Running a Single Application Remotely

RemoteApp is an RDP feature available on **Windows Server** (and Windows 10/11 Pro with a registry tweak) that lets you publish a specific application instead of a full desktop. The app opens in a window on your local machine — it looks like a local app but runs entirely on the remote server, including all its data and processing.

RemoteApp — applications running on server, displayed locally — server: 192.168.0.50

Instead of a full remote desktop, individual apps appear as windows alongside your local apps:

Excel 2021 (remote) | SAP Client (remote) | AutoCAD (remote) | VS 2022 (remote)

Each app appears in the taskbar, can be minimised, maximised, and Alt-Tabbed to — users can't tell it isn't running locally.

Enabling RemoteApp on Windows 10/11 Pro (registry tweak)

```
# On the HOST machine — run PowerShell as Administrator

# Enable RemoteApp on Windows 10/11 Pro (not available by default)
New-Item -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Serv
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Termi
  -Name "fDisabledAllowList" -Value 1

# Add an application to the allowed list
New-Item -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Serv
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Termi
  -Name "Path" -Value "C:\Windows\System32\notepad.exe"
```

```
# .rdp file for a RemoteApp session
full address:s:192.168.0.50
remoteapplicationmode:i:1                    # enable RemoteApp mode
```

```
remoteapplicationname:s:Notepad                  # display name
remoteapplicationprogram:s:notepad               # the app alias in allowed list
remoteapplicationcmdline:s:C:\docs\report.txt    # optional: open a specific file
username:s:philip
redirectclipboard:i:1
```

Microsoft Remote Desktop App (Store) vs mstsc

Microsoft publishes a modern **Remote Desktop** app on the Microsoft Store (free) alongside the classic `mstsc.exe`. They connect to the same RDP servers — the difference is in the management experience:

mstsc.exe

Built-in · Win32 · all Windows versions

The classic client. No installation needed, supports all .rdp file settings, full command-line control, every advanced option exposed. The right tool for power users and scripted connections.

✓ Always available — no install needed

✓ Full .rdp file and CLI support

✓ RemoteApp, admin session, all flags

⚠ UI feels dated on high-DPI displays

Microsoft Remote Desktop

Microsoft Store · modern UI · free

Modern, clean UI with a PC/workspace manager. Designed for managing multiple connections — add PCs, organise into groups, configure once and connect with one click. Better on touch screens and high-DPI monitors.

✓ Better multi-connection management

✓ Workspace/RemoteApp integration

✓ Cleaner high-DPI scaling

⚠ Fewer advanced settings exposed in UI

⚠ Doesn't use .rdp files directly

> **Recommendation:** use **mstsc** for scripted, automated, or advanced connections — particularly anything involving RemoteApp, custom .rdp files, or command-line flags. Use the **Store app** if you manage multiple RDP connections and want a cleaner UI for day-to-day interactive use.

Quick Reference — .rdp Key Cheat Sheet

Key	Values	Purpose
`full address:s:`	`host or host:port`	Target machine

Key	Values	Purpose
`username:s:`	`philip`	Login username
`screen mode id:i:`	`1=windowed, 2=fullscreen`	Window mode
`use multimon:i:`	`0 or 1`	Span all monitors
`dynamic resolution:i:`	`0 or 1`	Resize remote on window resize
`session bpp:i:`	`16, 24, 32`	Colour depth (bits per pixel)
`connection type:i:`	`1–7 (1=modem, 7=LAN)`	Network speed preset (sets experience flags)
`audiomode:i:`	`0=local, 1=remote, 2=none`	Where audio plays
`audiocapturemode:i:`	`0 or 1`	Redirect local microphone
`redirectclipboard:i:`	`0 or 1`	Clipboard sharing
`redirectdrives:i:`	`0 or 1`	Drive redirection
`drivestoredirect:s:`	`C:\; or * or DynamicDisks`	Which drives to share
`authentication level:i:`	`0=none, 1=warn, 2=NLA required`	NLA enforcement
`remoteapplicationmode:i:`	`0 or 1`	RemoteApp mode
`remoteapplicationprogram:s:`	`app alias or full path`	Which app to launch in RemoteApp mode

Next — Chapter 5: xRDP. Windows has RDP built in. Linux doesn't — but **xRDP** adds a full RDP server to Debian and Ubuntu, letting you connect from any RDP client (mstsc, Remmina, the Store app) to a Linux desktop. Chapter 5 covers installing xRDP, choosing a desktop session, connecting from Windows, and fixing the most common colour/display issues.

Chapter 5 — xRDP: Linux as an RDP Server

Windows ships with an RDP server built in. Linux does not — but **xRDP** is an open-source package that adds one. Once installed, any standard RDP client (`mstsc`, Remmina, the Microsoft Store app, xfreerdp) can connect to a Linux desktop session exactly as if it were a Windows machine, on port 3389.

> **Why xRDP instead of VNC?** xRDP uses the RDP protocol, which compresses well and supports clipboard, drive, and audio redirection natively. VNC sends raw pixel data and is typically slower over the same connection. If your client runs Windows, xRDP gives you a polished experience without any extra client software.

How xRDP Works

xRDP listens on port 3389, accepts the RDP connection, then spawns an **Xvnc** (or X11rdp) virtual display and launches your chosen desktop environment inside it. The desktop renders in that virtual display, and xRDP streams the output back to the client as standard RDP.

Choosing a Desktop Environment

xRDP works with most Linux desktop environments, but some perform better over RDP than others. For a remote connection, lighter is almost always better:

XFCE

Recommended

Lightweight, fast, stable over RDP. The de-facto standard recommendation for xRDP setups. Renders cleanly and uses very little CPU on the server.

MATE

Good choice

GNOME 2 fork. Slightly heavier than XFCE but familiar to Ubuntu users. Renders well over RDP and supports compositing.

LXDE / LXQt

Very light

Minimal resource use. Good for older hardware or very slow connections. Less polished than XFCE but responsive.

GNOME

Works, heavy

Full GNOME Shell works with xRDP but needs extra configuration to avoid black-screen issues. Uses significantly more CPU/RAM than lighter desktops.

KDE Plasma

Works, heavy

Fully functional but resource-intensive. Better suited to fast LAN connections. May show rendering glitches without GPU acceleration.

i3 / openbox

Power users

Tiling window managers work fine with xRDP and are extremely lightweight. Best for developers who prefer keyboard-driven workflows.

Installing xRDP on Debian / Ubuntu

debian — install xRDP + XFCE

```
philip@debian:~$ sudo apt update && sudo apt upgrade -y # install a lightweight desktop if not already present philip@debian:~$ sudo apt install -y xfce4 xfce4-goodies # install xrdp philip@debian:~$ sudo apt install -y xrdp # xrdp is added to the ssl-cert group so it can use the machine cert philip@debian:~$ sudo adduser xrdp ssl-cert # start and enable the service philip@debian:~$ sudo systemctl enable --now xrdp philip@debian:~$ sudo systemctl status xrdp ● xrdp.service - xrdp daemon Loaded: loaded (/lib/systemd/system/xrdp.service; enabled) Active: active (running) since Wed 2026-06-18 09:14:05 BST
```

Tell xRDP to use XFCE for your session

Create a `.xsession` file in the home directory of each user who will connect via RDP. This tells xRDP which desktop to launch:

debian — set session type

```
# write the session startup file philip@debian:~$ echo xfce4-session > ~/.xsession # alternative: use startxfce4 if xfce4-session is not found
```

```
philip@debian:~$ echo startxfce4 > ~/.xsession # for MATE instead of XFCE:
philip@debian:~$ echo mate-session > ~/.xsession # for GNOME (requires gnome-session): philip@debian:~$ echo gnome-session > ~/.xsession
```

Open port 3389 in UFW (if active)

debian — firewall

```
philip@debian:~$ sudo ufw allow 3389/tcp philip@debian:~$ sudo ufw allow from 192.168.0.0/24 to any port 3389 # prefer the second form — restrict to your LAN only
```

Do not expose port 3389 to the internet. xRDP has no built-in brute-force protection and RDP is a constant target for automated attacks. If you need remote access from outside your LAN, put xRDP behind an SSH tunnel (covered in Chapter 8) or a VPN. On the public internet, a default RDP port will be found and attacked within minutes.

Connecting from Windows (mstsc)

Open `mstsc`, enter the Linux machine's IP address, log in with your Linux username and password. That's it — the xRDP login screen appears first (you can leave Session as *Xorg* or the auto-detected default), then your XFCE desktop loads.

```
# Command-line shortcut from Windows
mstsc /v:192.168.0.24 /w:1920 /h:1080

# Or a saved .rdp file for the Linux box
full address:s:192.168.0.24
username:s:philip
screen mode id:i:2
authentication level:i:0     # xRDP has no NLA, so set to 0 (or 1 to warn)
redirectclipboard:i:1
audiomode:i:0
```

authentication level 0 vs 2: Windows may warn "The identity of the remote computer cannot be verified." This is expected — xRDP uses a self-signed certificate and does not support NLA

(CredSSP). Set `authentication level:i:0` in the .rdp file to suppress the warning, or click *Yes* once and tick *Don't ask again*.

Connecting from Another Linux Machine (Remmina)

linux client — xfreerdp to xRDP host

```
philip@laptop:~$ xfreerdp /v:192.168.0.24 /u:philip /p:password \ +clipboard
/dynamic-resolution /cert:ignore # /cert:ignore suppresses the self-signed cert
warning
```

In Remmina, create a new connection with Protocol set to **RDP**, enter the IP, username, and password. Under *Advanced*, set **Security** to *RDP (no NLA)* if the connection is refused or shows a certificate error.

Fixing Common xRDP Issues

Black / grey screen after login

The session started but the desktop environment didn't launch — missing or wrong .xsession file, or a desktop that isn't installed.

Fix: Check `~/.xsession` exists and contains the right command (`xfce4-session`). Install the desktop: `sudo apt install xfce4`.

Authentication / certificate warning

xRDP uses a self-signed TLS certificate. Windows shows a warning before connecting.

Fix: Set `authentication level:i:0` in the .rdp file, or click Yes on the prompt and tick "Don't ask again".

Colours look washed out / wrong

Colour depth mismatch. xRDP defaults to 24-bit; the client requests 32-bit.

Fix: In mstsc Display tab, set colour depth to 16-bit or 24-bit. In xfreerdp: `/bpp:24`. Also try disabling compositing in XFCE settings.

Session already logged in locally

If the user is already logged into the physical machine, xRDP creates a new separate session rather than attaching to the existing one.

Fix: Log out of the local session first, or use a different user account for RDP. xRDP cannot share an existing physical console session.

Connection refused / port not open

xRDP service not running, or UFW blocking port 3389.

Fix: `sudo systemctl start xrdp` and `sudo ufw allow 3389/tcp`. Check with `ss -tlnp | grep`

Clipboard not working

`xrdp-chansrv` process (the channel server for clipboard/audio) didn't start correctly.

Fix: Restart the session. Check `journalctl -u xrdp` for errors. Ensure xrdp is in the `ssl-cert`

`3389`.

group: `sudo adduser xrdp ssl-cert`.

Useful xRDP Diagnostics

debian — diagnose xRDP

```
# service status philip@debian:~$ sudo systemctl status xrdp xrdp-sesman # confirm xRDP is listening on port 3389 philip@debian:~$ ss -tlnp | grep 3389 LISTEN 0 10 0.0.0.0:3389 0.0.0.0:* users:(("xrdp",pid=1234,fd=12)) # live connection log — watch this while connecting from the client philip@debian:~$ sudo tail -f /var/log/xrdp.log philip@debian:~$ sudo tail -f /var/log/xrdp-sesman.log # check the per-session log if the desktop fails to start philip@debian:~$ ls ~/.xorgxrdp.*.log philip@debian:~$ cat ~/.xorgxrdp.10.log # installed xRDP version philip@debian:~$ xrdp --version xrdp: A Remote Desktop Protocol Server Version 0.9.21
```

Quick Reference — xRDP Configuration Files

```
# Main config — port, security, session limits
/etc/xrdp/xrdp.ini

# Session manager config — timeout, max sessions, X display range
/etc/xrdp/sesman.ini

# User session file — which desktop environment to launch
~/.xsession

# System-wide session file (fallback when no ~/.xsession)
/etc/xrdp/startwm.sh

# Connection and error logs
/var/log/xrdp.log
/var/log/xrdp-sesman.log

# Per-session Xorg log (NN = display number, typically 10)
~/.xorgxrdp.NN.log
```

Editing /etc/xrdp/startwm.sh lets you set the desktop globally for all users without needing a `.xsession` file in every home directory. Add `exec xfce4-session` before the final `test -x /etc/X11/Xsession ...` line. This is useful on a server where you control all the accounts.

Next — Chapter 6: VNC. xRDP sits on top of VNC internally, but VNC can also run directly — giving you access to a desktop without the RDP translation layer. Chapter 6 covers TigerVNC and RealVNC: setting up a server on Linux, connecting from any OS, and why you must always tunnel VNC over SSH rather than exposing it directly.

Chapter 6 — VNC: The Protocol, the Server, and the Tunnel

VNC (Virtual Network Computing) predates RDP and works on a simpler principle: it captures the screen as raw pixel data and sends it to the viewer. There is no server-side rendering intelligence — whatever appears on the display is streamed directly. This means VNC works with virtually any desktop environment and on any operating system, but it also means it uses more bandwidth than RDP for the same session quality.

VNC vs RDP — The Key Differences

RDP	VNC
Sends **rendering commands** — "draw this text at this position in this font"	Sends **pixel data** — a compressed copy of what's on screen
Creates a **new virtual session** — the physical display is unaffected	Can share the **actual physical display** — someone at the machine sees what you see
Encrypted by default (TLS)	**Unencrypted by default** — must tunnel over SSH
Better bandwidth efficiency — less data for same quality	Higher bandwidth use — more data, especially during motion
Native clipboard, drive, audio redirection	Clipboard sharing only (via extensions); no drive/audio natively
Windows-native; needs xRDP on Linux	Works on Linux, macOS, Windows — true cross-platform

> **When to prefer VNC over xRDP:** when you want to share or observe the *actual physical desktop* (great for remote support — you see exactly what the user sees), or when setting up xRDP

> is more trouble than it's worth on an unusual desktop environment. For headless server access, xRDP or SSH is almost always better.

VNC Display Numbers and Ports

VNC uses **display numbers** starting at `:1` (display `:0` is the physical screen). Each display maps to a TCP port: display number + 5900.

Display	Port	Typical use
`:0`	5900	Physical console — sharing the real screen
`:1`	5901	First virtual VNC display (default when you run vncserver)
`:2`	5902	Second virtual display (second user or second session)
`:3`	5903	Third virtual display, and so on

When connecting, you specify the host and display: `192.168.0.24:1` — or the full port: `192.168.0.24:5901`. Both mean the same thing.

Installing TigerVNC Server on Debian / Ubuntu

debian — install TigerVNC

```
philip@debian:~$ sudo apt update philip@debian:~$ sudo apt install -y tigervnc-standalone-server tigervnc-common # set a VNC password (stored separately from your login password) philip@debian:~$ vncpasswd Password: Verify: Would you like to enter a view-only password (y/n)? n # start a VNC server on display :1 with XFCE philip@debian:~$ vncserver :1 -geometry 1920x1080 -depth 24 New Xtigervnc server 'debian:1 (philip)' on port 5901 for display :1. # stop the server philip@debian:~$ vncserver -kill :1 Killing Xtigervnc process ID 3421... success
```

Configure the desktop environment for VNC

TigerVNC reads `~/.vnc/xstartup` to know which desktop to launch:

```
#!/bin/bash
# ~/.vnc/xstartup — launched by vncserver on startup

unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS

# launch XFCE
exec startxfce4

# --- alternatives ---
# exec mate-session
# exec gnome-session
# exec openbox-session
```

debian — make xstartup executable

```
philip@debian:~$ chmod +x ~/.vnc/xstartup # restart the server to pick up the new config philip@debian:~$ vncserver -kill :1 && vncserver :1 -geometry 1920x1080 -depth 24
```

Running VNC as a systemd Service

Starting the VNC server manually works for testing, but for a persistent setup you want it to start automatically on boot:

```
# /etc/systemd/system/vncserver@.service
# the @ allows multiple instances: vncserver@1.service, vncserver@2.service

[Unit]
Description=TigerVNC server — display %i
After=network.target

[Service]
Type=forking
User=philip
Group=philip
WorkingDirectory=/home/philip

PIDFile=/home/philip/.vnc/%H:%i.pid
```

```
ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill :%i > /dev/null 2>&1 || :'
ExecStart=/usr/bin/vncserver :%i -geometry 1920x1080 -depth 24 -localhost no
ExecStop=/usr/bin/vncserver -kill :%i

[Install]
WantedBy=multi-user.target
```

debian — enable the service

```
philip@debian:~$ sudo systemctl daemon-reload philip@debian:~$ sudo systemctl enable --now vncserver@1.service philip@debian:~$ sudo systemctl status vncserver@1.service
```

VNC is Unencrypted — Always Use an SSH Tunnel

VNC transmits everything in plain text by default — including your password (only weakly hashed) and every pixel of your screen. On a local trusted network this may be acceptable. Over the internet, or on any shared network, you must wrap VNC in an SSH tunnel. Never open VNC port 5901 to the internet directly.

The SSH tunnel approach

Forward the remote VNC port to a local port over SSH, then connect your VNC viewer to `localhost` rather than the remote IP. The traffic travels inside the encrypted SSH connection:

```
your machine localhost:5901
|
SSH tunnel encrypted SSH connection :22
|
remote server 127.0.0.1:5901 (VNC — only accessible locally)
```

```
# Step 1 — open the SSH tunnel (background, no shell)
ssh -fNL 5901:localhost:5901 philip@192.168.0.24
```

```
# Step 2 — connect your VNC viewer to the local end of the tunnel
# (address is localhost:5901, NOT the remote IP)
vncviewer localhost:5901
```

```
# One-liner — tunnel + viewer in the same command (Linux/macOS)
ssh -fNL 5901:localhost:5901 philip@192.168.0.24 && vncviewer localhost:5901

# Keep VNC server locked to localhost only (prevents direct connections)
# In your vncserver@.service, remove -localhost no and use:
vncserver :1 -geometry 1920x1080 -depth 24 -localhost yes
# Server will only accept connections from 127.0.0.1 — forces SSH tunnel
```

> **-localhost yes** is the right production setting. If VNC only listens on `127.0.0.1`, a direct connection from outside is impossible — the only way in is through the SSH tunnel. This means you never need to open port 5901 in your firewall at all.

VNC Viewer Clients

TigerVNC Viewer

Windows · Linux · macOS

The matching client for the TigerVNC server. Fast, lightweight, supports all TigerVNC extensions. Free and open-source. Best choice if your server is TigerVNC.

RealVNC Viewer

Windows · Linux · macOS · iOS · Android

Polished cross-platform client with optional cloud-relay (RealVNC Connect). Free tier available. The cloud relay avoids needing an SSH tunnel — useful but sends traffic through RealVNC's servers.

Remmina

Linux (GTK)

Supports both RDP and VNC in one app. Has a built-in SSH Tunnel tab so you can configure the tunnel inside the connection profile rather than running a separate ssh command.

KRDC / Vinagre

Linux (KDE / GNOME)

Desktop-environment native clients. KRDC for KDE, Vinagre for GNOME. Both support VNC and RDP. Good for occasional use; Remmina is more feature-rich for regular work.

Connecting from Windows with PuTTY + TigerVNC

If you're on Windows and prefer a GUI for the SSH tunnel, PuTTY can do it:

1. Open PuTTY → enter the server IP in *Host Name*
2. Go to **Connection → SSH → Tunnels**
3. Source port: `5901` · Destination: `localhost:5901` · click *Add*
4. Go back to *Session*, save the profile, then click *Open*
5. Once the SSH session is open, connect TigerVNC Viewer to `localhost:5901`

Common VNC Problems

Grey screen / no desktop

xstartup is missing, not executable, or contains the wrong command for the installed desktop.

Fix: Check `~/.vnc/xstartup` exists, is `chmod +x`, and references an installed desktop (e.g. `startxfce4`). Check `~/.vnc/debian:1.log`.

Authentication failed

Wrong VNC password — note this is different from your Linux login password.

Fix: Reset with `vncpasswd` then restart the server. The password file lives at `~/.vnc/passwd`.

Connection refused

VNC server not running, wrong port, or server started with `-localhost yes` but you're not going through the SSH tunnel.

Fix: Check `ss -tlnp | grep 590`. If tunnel is required, open the SSH tunnel first then connect to `localhost:5901`.

Slow / choppy display

VNC's pixel-data transport is bandwidth-hungry. Compositing effects multiply the data sent per frame.

Fix: Disable compositing in the desktop settings (XFCE: Window Manager Tweaks → Compositor → off). Use `-depth 16` or TigerVNC's `-ZlibLevel 9` compression flag.

Quick Reference — vncserver Flags

```
# start display :1, 1080p, 24-bit colour, locked to localhost
vncserver :1 -geometry 1920x1080 -depth 24 -localhost yes

# allow external connections (LAN only — still use SSH tunnel from internet)
vncserver :1 -geometry 1920x1080 -depth 24 -localhost no

# maximum zlib compression (slow CPU, saves bandwidth)
vncserver :1 -ZlibLevel 9

# list running VNC sessions
```

```
vncserver -list

# kill display :1
vncserver -kill :1

# kill all sessions for this user
vncserver -kill :*

# view the server log for display :1
cat ~/.vnc/$(hostname):1.log
```

Next — Chapter 7: NoMachine and Alternatives. RDP and VNC are the two foundational remote desktop protocols. Beyond them, a tier of modern alternatives offers better performance, easier setup, and built-in encryption. Chapter 7 covers **NoMachine** (NX protocol — often the fastest option on a LAN), **AnyDesk**, and **TeamViewer**: when each makes sense, and when you should stick with the open standards you already know.

Chapter 7 — NoMachine and the Alternatives

RDP and VNC are the two foundational remote desktop protocols — open standards supported by every operating system. But they were both designed in an era of fast local networks, and VNC in particular can feel sluggish over a broadband connection. A tier of modern tools has emerged that either builds on these protocols or replaces them entirely, trading openness for speed, simplicity, or both.

This chapter covers the four tools you're most likely to encounter: **NoMachine** (the performance king on a LAN), **AnyDesk** (fast and firewall-friendly), **TeamViewer** (the enterprise staple), and **Guacamole** (browser-based remote desktop — no client software required).

At a Glance — Which Tool Does What

Tool	Protocol	Needs relay server?	Encryption	Free tier	Best for
NoMachine	NX (proprietary)	No — direct P2P	TLS	Free for personal use	Fast LAN/WAN access, Linux desktops
AnyDesk	DeskRT (proprietary)	Optional (relay for NAT)	TLS 1.2 + AES-256	Free personal (non-commercial)	Remote support, firewall traversal
TeamViewer	Proprietary	Yes (by default)	TLS + AES-256	Free personal only	Enterprise support, unattended access
Guacamole	HTML5 (wraps RDP/VNC/SSH)	Self-hosted gateway	HTTPS	Fully open-source (Apache)	Browser access, zero client install

Tool	Protocol	Needs relay server?	Encryption	Free tier	Best for
RDP + xRDP	RDP (open standard)	No	TLS / NLA	Free	Windows hosts, structured access
VNC + SSH	RFB (open standard)	No	SSH tunnel required	Free	Physical screen sharing, cross-platform

NoMachine

NoMachine

NX protocol · direct connection · free for personal use · Windows / macOS / Linux / Raspberry Pi

NoMachine uses the **NX protocol** — originally developed by NoMachine (formerly based on X11 compression research) and now proprietary. It compresses X11 traffic far more aggressively than VNC and transmits rendering data rather than raw pixels, giving it RDP-like efficiency with VNC-like cross-platform reach. On a good LAN connection, it is often the fastest and smoothest remote desktop experience available for Linux hosts.

Strengths

- Noticeably faster than VNC, comparable to RDP
- Direct peer-to-peer — no relay server, no third-party cloud
- Can share the physical desktop (like VNC) or create a virtual session
- Audio, file transfer, USB sharing all built in
- Works on ARM — great for Raspberry Pi
- Free for personal use with no time limit

Limitations

- Proprietary — both ends must run NoMachine software
- No built-in NAT traversal — harder to reach from outside your LAN without port forwarding or VPN
- Commercial use requires a paid licence
- Less widely known than TeamViewer or AnyDesk in support scenarios

Installing NoMachine on Debian / Ubuntu

debian — install NoMachine server

```
# download the .deb from nomachine.com/download — check for the latest version
philip@debian:~$ wget
https://download.nomachine.com/download/8.x/Linux/nomachine_8.x.x_x86_64.deb
philip@debian:~$ sudo dpkg -i nomachine_8.x.x_x86_64.deb Selecting previously
unselected package nomachine. ... NoMachine was successfully installed. # service
starts automatically — check status philip@debian:~$ sudo /usr/NX/bin/nxserver --
status NX> 111 New connections to NoMachine server are enabled. # default port is
4000 (NX) — open in UFW if needed philip@debian:~$ sudo ufw allow from
192.168.0.0/24 to any port 4000
```

On the **client** side (Windows, macOS, another Linux machine), install the NoMachine player from the same download page. Enter the server's IP address and port 4000, log in with your Linux credentials, and choose whether to connect to the physical desktop or create a new virtual session.

> **NoMachine on Raspberry Pi:** Download the ARM build (`nomachine_x.x.x_armhf.deb` for Pi 3/4, `arm64.deb` for Pi 5). This turns any Pi with a desktop environment into a fully functional remote desktop server — great for accessing a headless Pi without setting up xRDP.

AnyDesk

AnyDesk

DeskRT codec · relay-assisted NAT traversal · free for personal non-commercial use

AnyDesk uses its own **DeskRT** video codec, optimised for desktop content (sharp edges, text, UI elements) rather than smooth video. The main selling point is its relay infrastructure: AnyDesk assigns each installation a unique 9-digit ID, and connections can be brokered through AnyDesk's relay servers when both sides are behind NAT — no port forwarding required. When a direct path exists, it prefers that instead.

Strengths

- Works through NAT and most firewalls without port forwarding
- Very low latency on good connections
- Tiny binary — no installation required (portable .exe on Windows)

Limitations

- Relay connections route through AnyDesk's infrastructure
- Commercial use requires a paid subscription

- Unattended access mode for servers and headless machines
- Free for personal non-commercial use

- AnyDesk has suffered phishing/social engineering attacks — employees granting access to "support" callers
- Closed source — you must trust AnyDesk's security claims

> **AnyDesk and remote scams:** AnyDesk is the tool of choice for "tech support" scammers — the caller asks the victim to install AnyDesk and read out their ID, then takes control of the machine. If you enable unattended access, use a strong password and be extremely careful who you give your AnyDesk ID to.

Installing AnyDesk on Linux

debian — install AnyDesk

```
# add AnyDesk repository philip@debian:~$ curl -fsSL
https://keys.anydesk.com/repos/DEB-GPG-KEY | \ sudo gpg --dearmor -o
/usr/share/keyrings/anydesk.gpg philip@debian:~$ echo "deb [signed-
by=/usr/share/keyrings/anydesk.gpg] \ http://deb.anydesk.com/ all main" | \ sudo
tee /etc/apt/sources.list.d/anydesk.list philip@debian:~$ sudo apt update && sudo
apt install -y anydesk # set an unattended-access password (allows connections
without user at the machine) philip@debian:~$ echo "your_password" | sudo anydesk
--set-password # get this machine's AnyDesk ID philip@debian:~$ anydesk --get-id
123 456 789
```

TeamViewer

TeamViewer

Enterprise remote support · relay-based · free for personal use · all platforms including mobile

TeamViewer is the most widely deployed remote support tool in enterprise environments, which means it's almost never blocked by corporate firewalls — IT departments recognise the traffic and whitelist it. Like AnyDesk it routes connections through its own relay infrastructure by default, assigning each installation a unique ID. It's the tool a non-technical user is most likely to have already heard of.

Strengths

Limitations

- Works through almost any corporate firewall or NAT
- Mobile apps — support someone on Android/iOS from a desktop
- Well-known — easy to ask a non-technical user to install it
- Unattended access, file transfer, remote printing, meeting mode
- Extensive audit logging in commercial tiers

- All traffic routes through TeamViewer's servers by default
- Aggressively detects "commercial use" and blocks free accounts
- Commercial pricing is expensive
- Heavier client than AnyDesk or NoMachine
- TeamViewer itself has suffered credential-stuffing breaches in the past

Apache Guacamole — Remote Desktop in a Browser

Apache Guacamole

HTML5 clientless gateway · wraps RDP / VNC / SSH · fully open-source · self-hosted

Guacamole is a **clientless remote desktop gateway**. You install it on a server; it proxies RDP, VNC, and SSH connections and serves the result as a web application. Users connect with any modern browser — no software to install on the client machine. This makes it ideal for jump hosts, lab access, and any scenario where you can't control what software is on the client device.

Strengths

- Zero client installation — any browser works
- Centralised access control — one place to manage who can reach which machine
- Supports RDP, VNC, and SSH from the same interface
- Fully open-source (Apache 2.0) and self-hosted
- Works well behind a reverse proxy (Nginx, Caddy, Cloudflare Tunnel)

Limitations

- Requires a dedicated server to run the Guacamole daemon
- Setup is more involved than single-machine tools
- Performance is limited by the server running guacd — not ideal for high-resolution video work
- Docker makes it easier but adds another layer to maintain

Guacamole with Docker — the quickest path

```
# docker-compose.yml — minimal Guacamole stack

services:
  guacd:
    image: guacamole/guacd
    restart: unless-stopped

  guacamole:
    image: guacamole/guacamole
    restart: unless-stopped
    ports:
      - "8080:8080"
    environment:
      GUACD_HOSTNAME: guacd
      GUACD_PORT: 4822
      # simplest auth — use a database in production
      GUACAMOLE_HOME: /etc/guacamole
    volumes:
      - ./guacamole-config:/etc/guacamole
    depends_on:
      - guacd
```

> **Guacamole behind Cloudflare Tunnel** is a powerful combo: Cloudflare handles the HTTPS and firewall traversal, Guacamole handles the protocol translation. You get browser-based access to any machine on your LAN, from anywhere, with no open ports. A natural extension of the Cloudflare Tunnel chapter from the Web Hosting course.

Choosing the Right Tool — Decision Guide

Fast LAN access to a Linux desktop

→ NoMachine

Best performance, direct connection, no relay, free. Install on both ends and connect by IP.

Connecting through NAT without port forwarding

→ AnyDesk or TeamViewer

Both use relay infrastructure to punch through NAT. AnyDesk is lighter; TeamViewer penetrates more corporate firewalls.

Supporting a non-technical user remotely

→ TeamViewer or AnyDesk

Both are easy for the other person to install. TeamViewer is more widely recognised. Ask them to read out the ID and password.

Accessing machines where you can't install client software

→ Apache Guacamole

Browser-based — any device with Chrome/Firefox works. Self-host for full control.

Accessing your own Linux server, privacy matters

→ xRDP or VNC over SSH

No third-party relay, no cloud dependency, no commercial licence concerns. Traffic stays between you and your server.

Windows-to-Windows on the same network

→ mstsc (built-in RDP)

Already installed, fast, encrypted, clipboard and drive sharing built in. No extra software needed.

Centralised access control for multiple machines

→ Apache Guacamole

Single web interface, user accounts, audit logs, supports RDP + VNC + SSH from one place.

Raspberry Pi remote desktop

→ NoMachine or xRDP

NoMachine has an ARM build and performs well. xRDP is lighter on CPU and works with any RDP client.

Next — Chapter 8: RDP and VNC Through Firewalls. The final chapter brings it all together: how to reach a remote desktop when you're outside your LAN and can't simply connect by IP. We'll cover SSH tunnelling (the safest method), Cloudflare Tunnel (zero open ports), performance tuning for slow or high-latency connections, and a checklist for building a reliable remote access setup.

Chapter 8 — RDP and VNC Through Firewalls

Everything so far has assumed you're on the same local network as the machine you're connecting to. The moment you step outside that network — whether working from a café, a hotel, or simply a different building — you hit two problems: **NAT** (your target machine doesn't have a public IP) and **firewalls** (ports 3389 and 5901 are blocked or dangerous to open). This chapter covers the practical solutions.

Three Approaches — Pick the Right One

SSH Tunnel

Recommended

Forward RDP or VNC through an existing SSH connection. No extra software, fully encrypted, works wherever SSH works. Requires port 22 (or a custom port) to be reachable.

Cloudflare Tunnel

Zero open ports

The tunnel dials out from your machine to Cloudflare. Nothing inbound needs to be open. Supports RDP natively via the WARP client or `cloudflared access`. Best for always-on remote access.

VPN

Full network access

Connect to a WireGuard or OpenVPN server on your home network — your device appears to be on the LAN. Then use RDP or VNC exactly as you would locally. Covered in full in the VPN course.

> **Don't open 3389 or 5901 to the internet directly.** Port 3389 is constantly scanned. Within minutes of opening it, automated tools will begin brute-forcing credentials. All three methods above avoid exposing these ports publicly.

Method 1 — SSH Tunnel for RDP

The pattern: forward the remote machine's RDP port (3389) to a local port on your machine, then point `mstsc` or Remmina at `localhost`. The RDP traffic travels inside the encrypted SSH connection.

```
your laptop localhost:13389
|
SSH tunnel (port 22) encrypted — internet
|
SSH jump host / home router port 22 open, 3389 closed
|
target Windows/Linux box 127.0.0.1:3389 (RDP)
```

```
# Tunnel — forward local port 13389 to the target's RDP port via the SSH host
# Use a high local port (13389) to avoid conflicts with a local RDP server
ssh -fNL 13389:TARGET_IP:3389 user@SSH_HOST

# Examples:
# Target IS the SSH host (same machine runs SSH and RDP)
ssh -fNL 13389:localhost:3389 philip@myhome.example.com

# Target is a different machine behind the SSH host (jump pattern)
ssh -fNL 13389:192.168.0.50:3389 philip@myhome.example.com

# Step 2 — connect mstsc to the local end of the tunnel
mstsc /v:localhost:13389
```

Persistent tunnel in ~/.ssh/config

```
# ~/.ssh/config — define the tunnel as part of the host alias
Host home-rdp
    HostName       myhome.example.com
    User           philip
    IdentityFile   ~/.ssh/id_home
    LocalForward   13389 192.168.0.50:3389    # or localhost:3389 if same host

# Usage: ssh -fN home-rdp   then   mstsc /v:localhost:13389
```

> **SSH on port 443:** Some networks block outbound port 22. Configure your SSH server to also listen on port 443 (`Port 443` in `/etc/ssh/sshd_config`). HTTPS traffic on 443 is almost never blocked. Combine with `Port 22` to keep the standard port working too.

Method 1b — SSH Tunnel for VNC

```
# Forward VNC display :1 (port 5901) through SSH
ssh -fNL 5901:localhost:5901 philip@myhome.example.com

# Then connect the VNC viewer to the local tunnel end
vncviewer localhost:5901          # Linux/macOS
# TigerVNC Viewer on Windows: enter   localhost:5901

# In Remmina — configure the SSH tunnel inside the connection profile:
# Protocol: VNC  →  SSH Tunnel tab  →  enable, enter SSH host details
# Remmina creates the tunnel automatically when you connect
```

Method 2 — Cloudflare Tunnel for RDP

Cloudflare Tunnel (`cloudflared`) creates an outbound-only connection from your server to Cloudflare's network. Nothing inbound is required. Cloudflare provides an RDP-specific access method through their Zero Trust product.

server — set up cloudflared RDP tunnel

```
# install cloudflared (if not already present from host1-host8 course)
philip@debian:~$ sudo apt install cloudflared # authenticate and create a named
tunnel philip@debian:~$ cloudflared tunnel login philip@debian:~$ cloudflared
tunnel create home-rdp # configure the tunnel to forward to RDP philip@debian:~$
cat ~/.cloudflared/config.yml tunnel: home-rdp credentials-file:
/home/philip/.cloudflared/<uuid>.json ingress: - hostname: rdp.example.com
service: rdp://localhost:3389 - service: http_status:404 # run the tunnel
philip@debian:~$ cloudflared tunnel run home-rdp
```

client — connect via cloudflared access

```
# on the client machine — install cloudflared # open a local listener that
forwards to the Cloudflare tunnel C:\Users\philip> cloudflared access rdp --
hostname rdp.example.com --url localhost:13389 # browser opens for Cloudflare
Zero Trust authentication # once authenticated, connect mstsc to the local
listener C:\Users\philip> mstsc /v:localhost:13389
```

> **Why Cloudflare Tunnel for RDP?** Port 3389 never needs to be open anywhere. Cloudflare's Zero Trust layer adds authentication before the RDP connection is even established — an attacker can't reach the RDP server at all. It also works from any network, including those that block SSH port 22.

Performance Tuning for Slow Connections

Remote desktop over a high-latency or low-bandwidth link feels sluggish because every screen update must make a round trip. The goal is to send less data per frame and tolerate more latency gracefully.

RDP (mstsc / xfreerdp) — what to disable first

Setting	mstsc .rdp key	xfreerdp flag	Saving
Colour depth → 16-bit	`session bpp:i:16`	`/bpp:16`	Large — halves pixel data
Disable wallpaper	`disable wallpaper:i:1`	`-wallpaper`	Large — removes background redraws
Disable font smoothing	`allow font smoothing:i:0`	`-fonts`	Moderate
Disable desktop composition	`allow desktop composition:i:0`	`-aero`	Moderate
Disable themes	`disable themes:i:1`	`-themes`	Moderate
Disable animations	`disable menu anims:i:1`	`-animations`	Moderate
Enable compression	`compression:i:1`	`/compression`	Moderate on slow links

```
# Slow-link .rdp profile — all bandwidth optimisations applied
full address:s:localhost:13389
session bpp:i:16
compression:i:1
disable wallpaper:i:1
disable themes:i:1
disable menu anims:i:1
```

```
allow font smoothing:i:0
allow desktop composition:i:0
connection type:i:2      # 2 = low-speed broadband preset
```

```
# xfreerdp slow-link one-liner
xfreerdp /v:localhost:13389 /u:philip /bpp:16 /compression \
  -wallpaper -themes -aero -fonts -animations \
  /dynamic-resolution +clipboard /cert:ignore
```

VNC — reducing bandwidth

```
# TigerVNC viewer — maximum compression, lower colour
vncviewer -compress 9 -quality 4 localhost:5901

# Server side — start with higher compression
vncserver :1 -ZlibLevel 9 -depth 16 -geometry 1280x720

# In xfce4 — disable compositing to reduce screen update frequency
# Settings Manager → Window Manager Tweaks → Compositor → uncheck "Enable displa
```

SSH tunnel compression

```
# -C enables gzip compression on the SSH tunnel itself
# Useful for VNC (pixel data compresses well); marginal gain for RDP (already co
ssh -fNL 5901:localhost:5901 -C philip@myhome.example.com
```

Diagnosing a Slow Remote Desktop Session

measuring your connection quality

```
# round-trip time to your home SSH host — the dominant factor philip@laptop:~$
ping myhome.example.com -c 10 rtt min/avg/max = 12.4/14.1/18.6 ms # excellent —
LAN-like rtt min/avg/max = 80/95/140 ms # acceptable — some lag visible rtt
min/avg/max = 250/300/400 ms # poor — use text mode if possible # measure
available bandwidth through the SSH tunnel philip@laptop:~$ ssh
philip@myhome.example.com 'dd if=/dev/zero bs=1M count=100' | \ pv > /dev/null
```

```
100MiB transferred in 8.3s — 12.0 MB/s (96 Mbit/s) # check CPU on the remote server — a maxed-out CPU causes lag independent of bandwidth philip@debian:~$ top -bn1 | head -5
```

Security Hardening Checklist

- [ ] **Never expose port 3389 or 5901 directly to the internet.** Use an SSH tunnel, VPN, or Cloudflare Tunnel instead.
- [ ] **Require key-based authentication for the SSH tunnel.** Disable password auth on the SSH server: `PasswordAuthentication no` in `/etc/ssh/sshd_config`.
- [ ] **Run fail2ban on the SSH server.** Even with key-only auth, fail2ban stops log spam and blocks scanners.
- [ ] **Use a non-standard SSH port (e.g. 2222 or 443).** Eliminates 99% of automated scan noise. Not security by obscurity — just noise reduction.
- [ ] **For VNC, start the server with `-localhost yes`.** VNC then only accepts connections from 127.0.0.1 — the SSH tunnel is the only way in, and port 5901 can stay closed in the firewall.
- [ ] **Use strong, unique passwords for RDP and VNC accounts.** RDP: ensure the Windows account has a password. VNC: set a password with `vncpasswd` separate from the system password.
- [ ] **Restrict RDP to specific users.** On Windows: *System Properties → Remote → Select Users*. Add only the accounts that need remote access.
- [ ] **Keep xRDP and TigerVNC updated.** Both have had security patches. `sudo apt upgrade xrdp tigervnc-standalone-server` periodically.
- [ ] **Review who has AnyDesk / TeamViewer unattended access.** Revoke access for old installations. Never share your ID publicly.
- [ ] **Check active sessions periodically.** On Windows: `query session` or Task Manager → Users tab. On Linux: `who` and `ss -tlnp | grep 590`.

Course Complete — Remote Desktop: RDP & VNC

You've covered the full spectrum of remote desktop technology — from the protocols and their trade-offs, through setting up Windows RDP, Linux xRDP, VNC, and modern alternatives, to reaching everything securely from outside your network. The tools you now know cover every scenario from a quick home-lab connection to a hardened production remote access setup.

8	3	6	10
Chapters	Protocols (RDP · VNC · NX)	Tools covered	Security checklist items