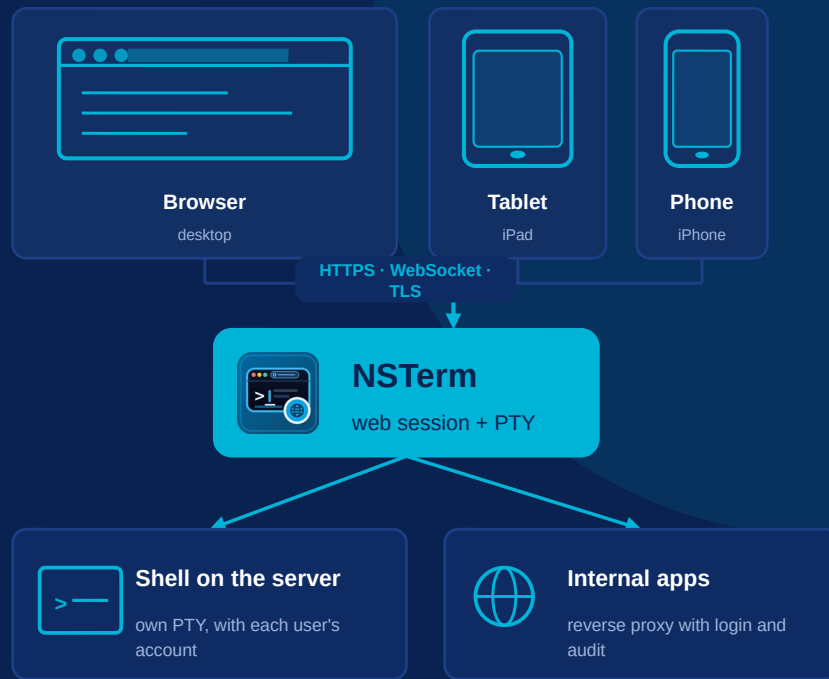


Your server terminal, in the browser. No SSH.

NSTerm publishes the shell as a web application: it opens in any browser, from a laptop, a tablet or a phone, over HTTPS. And internal consoles are published through the reverse proxy, with their own login and audit.

[SEE FEATURES →](#)

HTTPS only · No SSH keys · A single binary



connections by browser, tablet and phone

sessions · apps · groups · audit · proxy

SSH access to the server does not scale

Every operator carries their own SSH client, keys and tunnels: sessions cannot be audited and the internal console stays outside the access policy.

01 Keys per person

A keyring nobody maintains. Rotation becomes a quarter-end project.

02 No central identity

Everyone gets in as they can. No user, no role, no expiry.

03 No audit

The SSH login leaves no useful trace: nobody knows what ran or from where.

04 Loose consoles

The intranet is published on its own, bypassing the access policy.

- A clipboard and a text box

- No access control

- Untraceable consoles

QUÉ ES

The server shell, as a web application

NSTerm is a multi-session web terminal in a single binary: the administrator publishes applications (command, arguments, directory, environment and size), users open them in tabbed terminals over WebSocket and every process is tracked as a session.

- ✓ xterm terminals in the browser, with tabs and reconnection
- ✓ Sessions with PID, user, IP, country, browser and child processes
- ✓ The command runs on the server, with the user the administrator picks
- ✓ Internal consoles published by reverse proxy, with login and RBAC

0

SSH keys to manage

1

binary, no dependencies

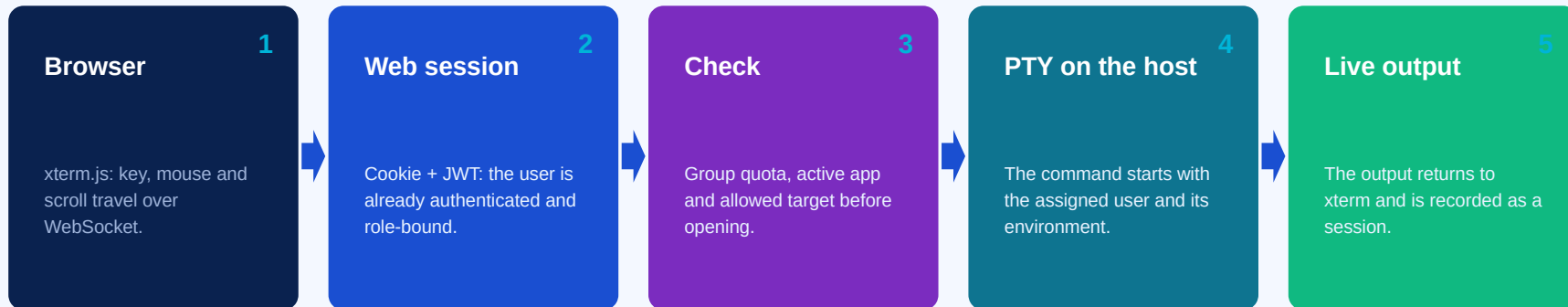
HTTPS

the only network need

3

kinds of device

How a keystroke travels, step by step



⚠ The session lives on the server: if the browser closes, the process keeps running and can be picked up later.

HTTPS and WSS only

No shell on the client

Auditable session

No SSH: HTTPS is enough

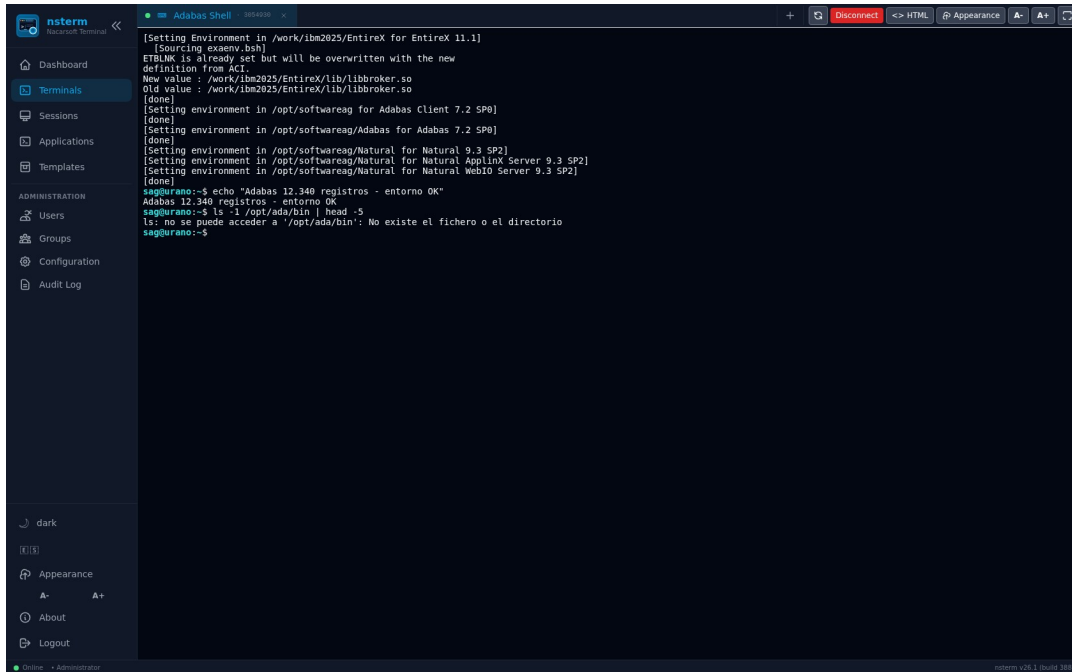
No sshd, no keys, no bastion, no tunnels. NTerm speaks WebSocket over TLS and the command runs on the server itself.

Access	SSH client + private key	Browser with a session cookie
Identity	The user's public key	Panel user, group and role
Audit	At best, auth.log	Log per session, user and IP
Quotas	None	Max sessions per user and group
Internal consoles	Tunnels and port forwarding	Reverse proxy with RBAC
For the user	Install and configure a client	Open a URL

The only network requirement is a reverse proxy with HTTPS on port 11030.

CONEXIONES

From the desktop browser



- xterm.js with the server character set and colours
- Tabs: several terminals at once, with name and PID
- Reconnection: reload the page and the session is picked up
- Full screen, font sizing and HTML mirror
- Shortcuts and Natural function keys (PF) when the app asks for them

No plugin

No PuTTY Java

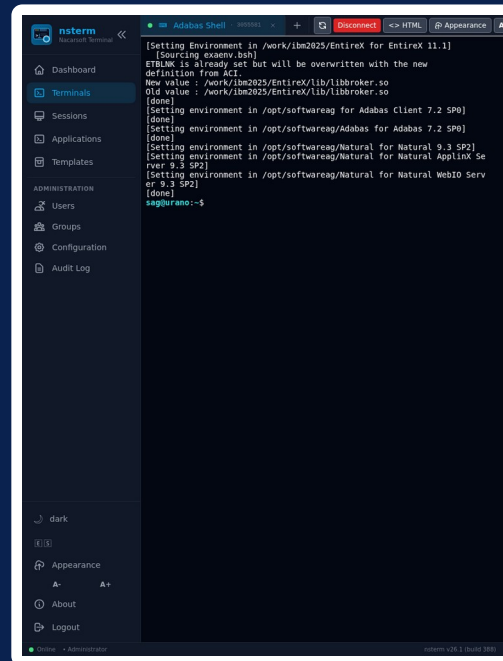
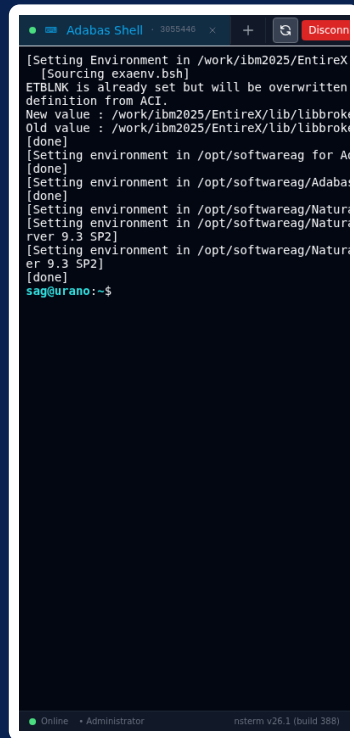
Bookmarks and
favourites

A real terminal in the browser, with the session PID on the tab.

Also from the tablet and the phone

- The same session, with the layout adapted to the screen width
- On-screen keyboard with the keys the app needs
- Long press and text selection over the command output
- Ideal for one-off queries and on-call duty away from the desk

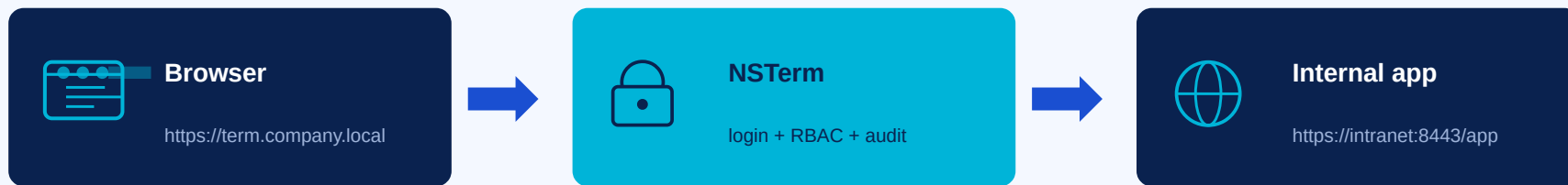
The very same link works on the PC, the tablet and the phone.



Tablet: 834 px, a comfortable working width.

Reverse proxy: publish your internal consoles

Beyond terminals, NTerm publishes internal web applications: it exposes an http/https URL through NTerm itself, inheriting the login, the per-group RBAC and the audit.



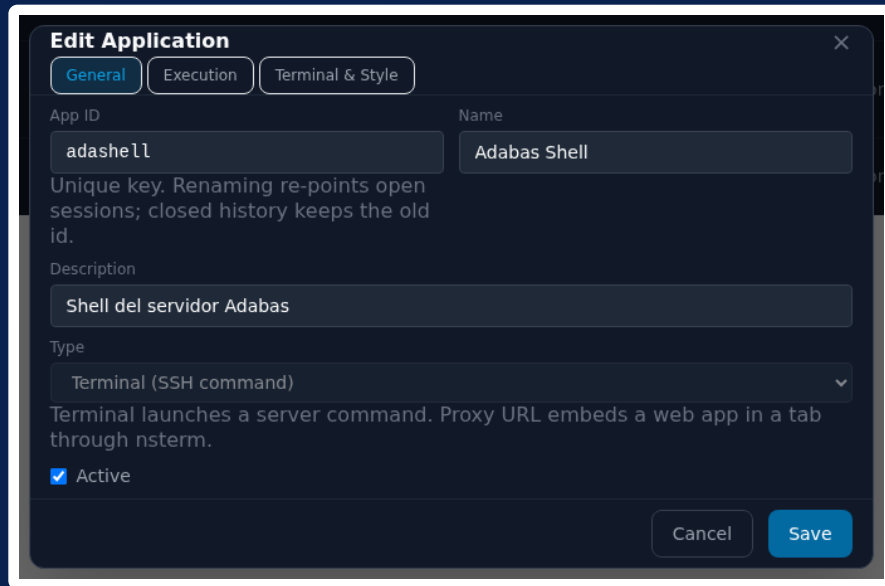
- Only admin-defined URLs: the user never types one
- Host allowlist, private-IP and cloud-metadata blocking
- Link rewriting and WebSocket support on the backend
- Per-application body limit and response timeout

The internal console leaves the network and goes through the very same login as the terminal.

Applications: the admin decides what is published

- Command, arguments, working directory and environment
- Terminal size (cols/rows) or fit to the container
- Colour scheme, icon, logo and application colour
- System user it runs as: fixed or the connected user's account
- Session timeout, HTML design and Natural keys

A user only sees the applications assigned to their groups.



The screenshot shows a dark-themed 'Edit Application' dialog box with a close button (X) in the top right corner. It features three tabs: 'General' (selected), 'Execution', and 'Terminal & Style'. The 'General' tab contains the following fields:

- App ID:** A text field containing 'adashell'.
- Name:** A text field containing 'Adabas Shell'.
- Description:** A text area containing 'Shell del servidor Adabas'.
- Type:** A dropdown menu currently showing 'Terminal (SSH command)'.
- Active:** A checkbox that is checked.

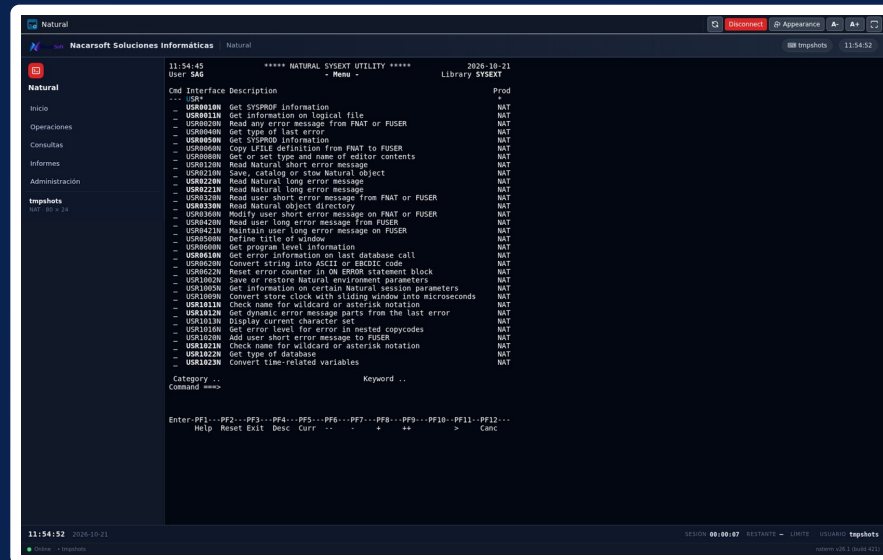
Below the 'Type' dropdown, there is explanatory text: 'Terminal launches a server command. Proxy URL embeds a web app in a tab through nsterm.' At the bottom right of the dialog are 'Cancel' and 'Save' buttons.

The application editor: id, name, description, type and state.

Templates: the terminal with a web application look

- Header, side containers and status bar with your own HTML and CSS
- Corporate logo and name, taken from the server configuration
- PF1..PF12 buttons that send the key to the session
- Live values: clock, date, session time and remaining time
- The CSS runs isolated: it cannot break the NSTerm interface

Eleven example templates are created on install: assign the one you want to each application.



The same application, with a corporate header, side menu and a status bar with the clock.

HORARIOS

Schedules: the application is only available in its window

- Schedule templates: morning, afternoon, night, weekend
- Outside the window the app shows as Closed on the dashboard
- The server blocks it: it cannot be opened bypassing the panel
- Window per day, several windows and night shifts
- Administrators are subject to the schedule as well

Name	Days	Windows	State	Actions
Fines de semana Todo el día, sábado y domingo	S-D	00:00-24:00	Closed	 
Horario laboral De 08:00 a 12:00 y de 16:00 a 18:00, de lunes a viernes	L-V	08:00-12:00, 16:00-18:00	Open	 
Mañana De 08:00 a 14:00, de lunes a viernes	L-V	08:00-14:00	Open	 
Noche De 22:00 a 06:00, de lunes a viernes (cruza medianoche)	L-V	22:00-06:00	Closed	 
Tarde De 16:00 a 20:00, de lunes a viernes	L-V	16:00-20:00	Closed	 

The available schedule templates and their current state.

Console access becomes office hours, instead of always being open.

Every session is a process, and it is recorded

- Application, user, PID, start and duration of every session
- IP, country and browser of whoever opened it
- Child processes: what the command has spawned
- CPU, memory and the configured session timeout
- Kill the session asking for a reason, which goes to the log

App	User	PID	Started	Duration	Timeout	CPU	MEM	IP	Country	Browser	Actions
 adashell	admin	3855774	22:36:08	5s	⌚ 4h · 3h 59m 51s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 adashell	admin	3855581	22:36:03	10s	⌚ 4h · 3h 59m 46s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 adashell	admin	3855446	22:35:58	15s	⌚ 4h · 3h 59m 41s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 naturalsh	admin	3855227	22:35:54	19s	⌚ 2h · 1h 59m 37s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 adashell	admin	3854938	22:35:45	28s	⌚ 4h · 3h 59m 28s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 adashell	admin	3854939	22:35:13	1m 0s	⌚ 4h · 3h 58m 56s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 adashell	admin	3853878	22:35:08	1m 5s	⌚ 4h · 3h 58m 51s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 adashell	admin	3853737	22:35:03	1m 10s	⌚ 4h · 3h 58m 46s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 naturalsh	admin	3853545	22:34:59	1m 14s	⌚ 2h · 1h 58m 42s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	
 adashell	admin	3853356	22:34:51	1m 22s	⌚ 4h · 3h 58m 34s	–	6 M	127.0.0.1	Local	Chrome 144 · Linux	

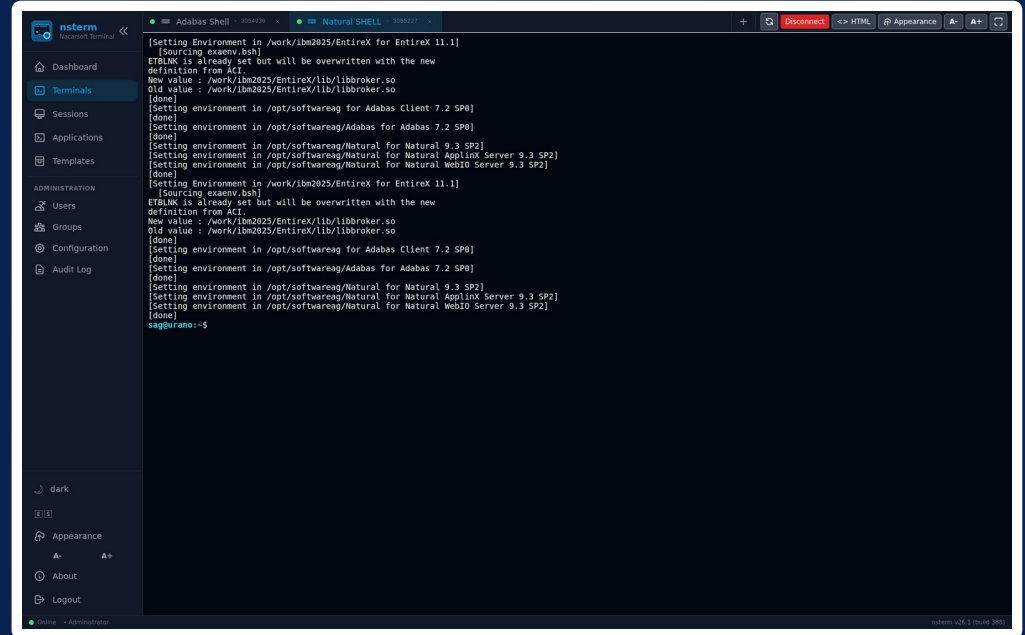
Sessions: who runs what, from where and since when.

The table updates live over WebSocket: no need to reload.

MULTITERMINAL

Several terminals, one window

- A new tab without losing the previous session
- Each tab with its own PID and application name
- Automatic reconnection after a network drop
- The session state, visible in the tab bar



Tabs: one terminal per session, all in the same window.

Users, groups and session quotas

- A user belongs to groups; the group defines the role
- Each group has its applications assigned
- Quota of simultaneous sessions per user and application
- Past the quota, the terminal warns and opens no session
- User administration: alias, email and state

Name	Description	Role	Applications	Created	Actions
admins	Administrators	Admin	—	2026-10-19	Apps Delete
guardia	Guardia de produccion	User	Natural SHELX	2026-10-19	Apps Delete
operadores	Operadores de terminal	User	Adabas Shell Console Interna	2026-10-19	Apps Delete
users	Default user group	User	—	2026-10-19	Apps Delete

Groups: role, assigned apps and the maximum sessions per user.

The same application can live in several groups, with different quotas.

Who opened what, and it is recorded

- Logins, app creations, group changes and session endings
- Mandatory reason when closing somebody else's session
- Optional LDAP or Active Directory, alongside local users
- Password policies: expiry, early warning and history
- Session token with configurable expiry and HttpOnly cookie

Time	User	Action	Detail	IP	Country
2026-10-19 22:36:08	admin	terminal_open	adashell	127.0.0.1	Local
2026-10-19 22:36:03	admin	terminal_open	adashell	127.0.0.1	Local
2026-10-19 22:35:58	admin	terminal_open	adashell	127.0.0.1	Local
2026-10-19 22:35:54	admin	terminal_open	naturalsh	127.0.0.1	Local
2026-10-19 22:35:45	admin	terminal_open	adashell	127.0.0.1	Local
2026-10-19 22:35:39	#0	login	admin	127.0.0.1	Local
2026-10-19 22:35:13	admin	terminal_open	adashell	127.0.0.1	Local
2026-10-19 22:35:08	admin	terminal_open	adashell	127.0.0.1	Local
2026-10-19 22:35:03	admin	terminal_open	adashell	127.0.0.1	Local
2026-10-19 22:34:59	admin	terminal_open	naturalsh	127.0.0.1	Local
2026-10-19 22:34:51	admin	terminal_open	adashell	127.0.0.1	Local
2026-10-19 22:34:45	#0	login	admin	127.0.0.1	Local
2026-10-19 22:34:45	admin	theme_change	dark	127.0.0.1	Local
2026-10-19 18:46:39	admin	terminal_close	session=63 app=adashell pid=2593451 reason=idle timeout (no clients)	127.0.0.1	Local
2026-10-19 18:46:34	admin	terminal_close	session=62 app=adashell pid=2593296 reason=idle timeout (no clients)	127.0.0.1	Local

Audit log: logins, applications, groups and session endings.

Machine-signed license and a 30-day trial.

From zero to a web terminal, in three steps

1

Create the environment

`nsterm init /opt/nsterm`: it creates `config.yaml`, the SQLite database, the logs and the admin user.

2

Publish the reverse proxy

Terminate TLS at the proxy and forward to `https://server:11030`. The server itself is not touched.

3

Publish the shell

Create the application (command, args, directory and environment) and assign it to the groups that need it.

```
nsterm init /opt/nsterm → nsterm start --config /opt/nsterm →  
https://term.company.local
```

REQUIREMENTS

- Linux (RHEL 8 / 9)
- A single binary, no dependencies
- SQLite for state and audit
- Reverse proxy with HTTPS in production
- Port 11030 (configurable)

No SSH: no need to open port 22, deploy keys or touch the `sshd` configuration.

FREQUENTLY ASKED

The questions we hear most

Is it really true that no SSH is needed?

Yes. The browser talks HTTPS and WebSocket with NSTerm and the command runs on the server with its own PTY. sshd is not involved.

Who runs the command?

The admin decides: a fixed system user or the Linux account mapped to the connected user. NSTerm verifies the sudoers permissions.

Can I publish an internal website?

Yes: create a Proxy URL application with its target and assign it to groups. Users open it with their session and with no direct access to the URL.

What if the browser closes?

The session stays alive on the server. When you come back, the tab is restored and re-attached to the same process, with its PID.

How is usage limited?

Each group has a maximum of simultaneous sessions per user and application. Past it, the terminal warns and opens no new session.

Can it integrate with Active Directory?

Yes: LDAP/AD is optional and lives alongside local users. The user is mapped to their Linux account for the PTY.

Your server, in the browser and with its own audit trail

Try NSTerm with your environment: we set it up with your applications
and watch it run.

[Guided demo](#)

[Try it in your environment](#)

[Let's talk about your case](#)

nacarsoft.com · support@nacarsoft.com

