



NOVANAS / PROJECT DOCUMENTATION

ADMINISTRATOR + DEVELOPER GUIDE

NovaNAS

Storage. Applications.
One place to manage them.

Installation, everyday operations, architecture,
updates, recovery and release engineering.

30 OCTOBER 2026

Based on the audited project and current installer release.

Contents

Practical guidance for a new installation, existing administration and project development.

01	Project overview	3
02	Installation and first boot	4
03	Disks, RAID and pools	5
04	Shares, users and snapshots	6
05	File Manager and link sharing	7
06	Docker Manager	8
07	Virtual Machines	9
08	Surveillance Station	10
09	Antigravity AI and Terminal	11
10	Updates and automatic recovery	12
11	Networking and notifications	13
12	Security, backups and recovery	14
13	System architecture	15
14	Configuration and source map	16
15	Development and release workflow	17
16	Operational checks and troubleshooting	18
17	Release baseline and verification	19

How to use this guide

Start with installation and storage for a new NAS. Use the individual workspace chapters for daily operations. Architecture, source mapping and release workflow are intended for maintainers. Procedures describe supported workflows; they do not replace reading an operation's actual plan or confirmation.

Project overview

A browser-managed storage and application platform built on Debian.

What NovaNAS provides

NovaNAS brings storage management, file access, containers, virtual machines, surveillance and administrative tools into one web interface. Its installer targets Debian 13 on 64-bit x86 hardware. The installed system is headless: administration happens from another device's browser.

Area	Main workspace
Monitor	Dashboard: CPU, memory, network, temperatures and supported GPU metrics.
Store	Disks, RAID, Pools, Shares, Users and read-only Btrfs Snapshots.
Access	File Manager: local storage, external SMB/NFS locations and public sharing links.
Run	Docker Manager and Virtual Machines, including protected updates and offline cloning.
Record	Surveillance Station: camera streams, recording, events, retention and Drive settings.
Administer	Settings, update checks, notifications, networking, Terminal and Antigravity AI.

Hardware planning

Use a separate system disk where practical, storage devices suitable for the intended RAID layout, and a supported network adapter. Virtual machines require CPU virtualization support and enough uncommitted RAM and disk capacity. Camera encoding and multiple workloads can require substantially more CPU, memory or a compatible GPU.

No universal minimum RAM, throughput guarantee or hardware compatibility certification is established by this project guide. Validate the actual controller, network adapter, firmware and workload before relying on the system.

Release baseline

This guide describes the audited 30 October 2026 sources and installer repack. It is a project and operations guide, not a claim that every feature has been tested on every machine.

Installation and first boot

Install from clean media and create the administrator during installation.

1 · Prepare and verify the media

1. Download or copy the latest ISO and its adjacent `.iso.sha256` file.
2. In the directory containing both files, run the checksum command below. The result must say OK.
3. Write the ISO using an image-writing utility, or attach it as a VM's bootable optical media.
4. Back up the intended target disk and review disk identities before starting installation.

```
sha256sum -c novanas-installer-amd64-20261030-185302.iso.sha256
```

2 · Run the installer

The hybrid ISO contains BIOS and UEFI boot entries. The default route launches the NovaNAS-branded Calamares installer. Choose the installation disk, review partitioning and the erase summary, set a hostname, and create a regular administrator account. Debian Installer remains a fallback under advanced boot options.

3 · Complete first boot

Remove the installation media and boot the installed system. First boot configures the installer-created administrator, storage helpers and service permissions, developer-tool packaging, SSH host keys and network discovery. The temporary live-session account is removed from the installed system.

Read the LAN address or hostname shown on the console. Open `http://<nas-ip>` or `http://<hostname>.local` from the same network. Sign in with the administrator account you created.

Before writing data

Confirm the target disks and networking, then create or import storage through the appropriate workspaces. A completed OS installation does not establish an initialized or healthy storage array.

Secure Boot and individual hardware combinations still require validation. The exact current repack passed packaging checks but has not been fresh-installed or boot-tested.

Disks, RAID and pools

Build the storage layout first; expose shares and application data afterward.

Recommended workflow

1. Use **Disks** to identify physical devices, inspect their reported health and confirm which devices are already in use.
2. Use **RAID** to plan the desired conventional or NovaNAS Hybrid RAID layout. Review available capacity, protection space and unused space.
3. Confirm the intended members before creation or expansion. Destructive operations must target only disks you intend to allocate.
4. Use **Pools** to manage the storage presented to shares and applications. Validate mount status before creating dependent workloads.

NovaNAS Hybrid RAID planning

NHR divides devices into aligned size tiers and makes each usable tier redundant independently. Capacity that cannot satisfy the selected protection level remains unallocated.

Profile	Minimum	Planning behavior
NHR-1	2 disks	One-disk fault tolerance. Two-member tiers use mirroring; tiers with three or more members use RAID5.
NHR-2	4 disks	Two-disk fault tolerance. Four-or-more-member tiers use RAID6; eligible three-member tiers use three copies.

Example: an NHR-1 layout with nominal 4, 8 and 8 TB devices can use a common three-disk tier plus a mirrored upper tier. Actual displayed capacity depends on alignment, exact device sizes, metadata and the chosen filesystem.

Expansion

Expand RAID Array exposes a plan and controlled expansion workflow. Read the proposed changes before applying them. An array reshape can take substantial time and affects the entire storage stack; monitor its status and maintain an independent backup during the operation.

RAID is availability, not a backup

Redundancy can tolerate the expected member failures; it does not protect against deletion, ransomware, application corruption, theft or loss of the whole machine.

Shares, users and snapshots

Organize data access and keep recovery points on supported Btrfs shares.

Shares and permissions

Create shares on the intended storage and assign access through the Shares and Users workspaces. Use dedicated users where separate file access is required. Check read/write permissions with the intended client account rather than testing only as the administrator.

The web administrator selected during installation is separate from the restricted novanas service account. Storage users and share permissions do not imply unrestricted web-management or root access. Do not give an application broader share access than it needs.

Read-only Btrfs snapshots

The Snapshots workspace creates read-only snapshots of shares and displays snapshot history, creation time, Btrfs identity and read-only status. Automatic snapshots let you choose an interval, selected shares and how many recent snapshots to retain.

1. Select a share and create a manual snapshot before a substantial data change.
2. Configure automatic snapshots for the shares that need recurring recovery points.
3. Review retention carefully: it applies to matching snapshots of the selected shares.
4. Confirm an independent backup exists before deleting snapshots you may need for recovery.

Recovery mechanism	Protects against	Does not provide
Read-only share snapshot	Some accidental data changes; a point-in-time view of one share.	An off-device copy or guaranteed application-consistent database backup.
RAID redundancy	Expected physical member failures.	Historical file versions or protection against deletion.
Independent backup	Loss beyond the local storage stack, when tested and maintained.	Automatic recovery unless a restore procedure is practiced.

Do not assume a one-click whole-system restore exists because share snapshots are available. Plan file recovery and application/database recovery separately.

File Manager and link sharing

Browse storage, move data and create links using the address visitors can reach.

Everyday file operations

File Manager provides storage navigation, breadcrumbs, list/grid views, sorting and contextual file actions. Transfer progress can show transferred size, speed and estimated time remaining. External SMB/NFS locations can be mounted through the supported workflow; write access depends on the remote account and server permissions.

Create a sharing link

1. Open NovaNAS through the address the recipient should use. For public sharing, use the intended HTTPS domain.
2. Select a file or folder and open its sharing dialog.
3. Set the available restrictions, such as a password, expiry or download limit, then create and copy the link.
4. Test the link in a private browser window or a recipient device. For a public link, also test from outside the local network.
5. Revoke the link when it is no longer needed.

How the address is chosen

New links use the current browser origin: scheme, domain or IP address, and any explicit port. A link created while connected through a local IP remains a local-IP link. A domain is not made publicly reachable merely by appearing in the URL; DNS, HTTPS and reverse-proxy routing must also work.

The public share route is `/s/`. The installed proxy routes it and link management to the worker on loopback port 8012. Folder sharing packages the selected content as a ZIP for download.

Sharing checks

If a link works locally but fails remotely, check DNS, certificates and proxy routing first. A password prompt is separate from NovaNAS administrator login; do not send administrator credentials to a recipient.

The updated dialog preserves its selected target and folder listing while opening, submitting and closing. Password inputs are isolated from folder search so browser autofill does not unexpectedly filter the visible files.

Docker Manager

Manage application containers and update them through their own workflow.

Container lifecycle

Use Docker Manager to inspect containers, manage supported lifecycle actions and apply image updates. Keep persistent application data in intentional volumes or bind mounts. Record which paths contain configuration, databases, uploads and secrets.

Compose-managed services and standalone containers follow different update paths. The manager selects the appropriate strategy from their configuration rather than treating every container as an isolated image replacement.

Protected updates

1. Review the container or stack and back up important application data.
2. Start its update in **Docker Manager**.
3. The independent worker retains the previous image/configuration, applies the supported update strategy and validates the result.
4. Inspect the job and any warnings. Failed validation attempts to restore the retained configuration and image.

The protection applies by default to supported existing and future containers, including standalone, managed Compose and local-build Compose workflows. Persistent job records support recovery after an interrupted operation.

Validation	What it establishes
Docker HEALTHCHECK present	The worker checks the container's reported health and startup stability.
No HEALTHCHECK	Only startup stability is verified. This does not prove the application is functioning correctly.
Rollback completed	The retained image/configuration was restored and checked within the worker's supported scope.

Application data is not rewound

Volumes, bind-mounted files and database migrations are not reverted with the image. A new application may alter its data before failing. Keep compatible backups and review migration requirements before updating.

Docker application updates are separate from **NovaNAS Updates**. Updating Debian's Docker engine package is not the same as updating containers.

Virtual Machines

Create guests, attach installation media and clone supported powered-off machines.

Create and install a guest

1. Confirm host virtualization is available and reserve sufficient RAM, CPU and storage.
2. Upload or import an installation ISO using the Virtual Machines workspace.
3. Create a VM with appropriate disk, firmware and network settings.
4. Attach the ISO, start the VM and complete the guest OS installation through its console.
5. Eject installation media when the guest no longer needs it.

Power and networking

Prefer a graceful guest shutdown. Force shutdown is equivalent to cutting power and can corrupt unsaved work or guest filesystems. Network bridge changes can disconnect the management connection; use the guarded bridge workflow and its confirmation mechanism.

Clone a VM

Shut down the source VM, then choose **Clone VM**. The full clone creates independent supported disk copies, new hypervisor UUID/MAC identities and independent firmware variables where applicable. The clone is left powered off. Background progress and recent jobs remain available after the dialog closes.

The clone contains the source guest's files, settings and accounts. New hypervisor identities do not automatically change the guest hostname, application identity, static IP address, SSH keys or licensed-software identity. Review those inside the guest before using both machines on the same network.

Supported scope

Unsupported TPM, encrypted or shared/block-disk configurations are refused rather than silently copied. Review the reported reason and use a guest-aware or application-aware migration procedure if necessary.

Guest backups

A clone is useful for duplication but is not a complete backup policy. Coordinate databases and applications inside the guest, keep an independent copy, and verify that the guest can boot and its services can restore.

Surveillance Station

Monitor cameras and manage recording storage with explicit retention safeguards.

Camera and recording workflow

1. Add supported cameras using the discovery or camera configuration tools.
2. Verify a live stream before enabling persistent recording.
3. Select the intended recording location and confirm storage access.
4. Configure recording, event/motion options and hardware acceleration as supported by the camera and host.
5. Check live playback, recorded playback, free capacity and retention results after changes.

Playback and encoding

Live viewing uses HLS-oriented stream handling. Camera-specific encoding can override the global hardware-acceleration preference. A selected GPU option does not establish that a stream is using it: inspect the active encoder and recent stream output when diagnosing playback or load.

Safe retention cleanup

Retention depends on the recording policy and trusted folder/provenance metadata. Cleanup validates the root and camera folders, protects historical baseline files and enforces file identity and safety checks before deletion. An identity mismatch should be investigated, not bypassed by deleting guard metadata.

The current Btrfs repair uses persistent filesystem and subvolume UUIDs plus inode identity. Kernel-assigned device numbers can change between mounts; they are still useful within one cleanup run, but are no longer the only persistent identity for enrolled Btrfs recording storage.

Before changing recording storage

Back up configuration and verify the new filesystem, subvolume and folder identities. Do not transplant another installation's trusted cutoff, baseline, registry or recording-session files.

Google Drive

Drive upload and existing-recording backfill are separate settings. If authorization expires, reconnect the account through the provided flow and test a new upload. Saved client details or a stored refresh token do not, by themselves, prove that uploads currently work.

Antigravity AI and Terminal

Use the administrator's tools while keeping account connection and privileges clear.

Connect Antigravity

1. Sign in with the administrator account created for this installation.
2. Open **Antigravity AI** and choose **Connect account**.
3. Follow the displayed authorization link and code using your own Google account.
4. Complete the connection in NovaNAS, then verify a simple request before relying on administrative actions.

The installer includes Antigravity and Codex CLI binaries. It does not include a previously signed-in user, OAuth credentials, saved conversations, Hermes, Ollama or local model libraries from the build host.

Tool access

The current Antigravity interface reports automatic tool approval. Treat access to it as administrative access: requests can cause real commands or configuration changes within the account's effective privileges. Restrict the management interface and do not paste unknown instructions into an administrative session without understanding their effect.

Integrated terminal

The terminal connects over WebSocket and runs the signed-in administrator's shell through a constrained helper. Reconnect can reuse a detached, live shell belonging to the same account. A disconnected browser is not necessarily a terminated shell; inspect the session before starting a duplicate long-running job.

Symptom	First check
Terminal closes with code 1006	Proxy WebSocket headers, idle timeouts, session state and backend reachability.
Antigravity cannot connect	Account authorization, backend/CLI availability and network access.
CLI command appears stalled	Its process and task output; avoid repeatedly starting the same operation.

Backend restarts interrupt sessions

Terminal and AI work may share the backend's systemd service lifetime. Schedule backend restarts from an independent session and assume an active browser terminal can be disconnected or stopped.

Updates and automatic recovery

Check on a weekly schedule; install deliberately; understand exactly what rollback covers.

Package-management workflow

1. Open the Updates workspace.
2. Choose **Check Available Updates**. NovaNAS refreshes repositories, waits for completion and then lists current package upgrades.
3. Review the available packages and installation log.
4. Choose **Install Updates** when ready. Services may restart during installation.
5. Review completion, recovery messages and the reboot-required indicator.

The weekly-check toggle controls automatic package and managed CLI checks every seven days. Turning it off stops that automatic check schedule. A manual check remains available. Notifications depend on the notification settings. Scheduled checks do not automatically install the listed packages.

Package rollback boundaries

Covered behavior	Important boundary
Retain old package versions and package conffiles	An unavailable previous archive blocks a protected installation.
Post-update package/service checks and automatic restore attempt	A passing health check cannot prove every application workflow is correct.
Recovery journal and watchdog for interrupted operations	An OS that cannot boot needs a separate recovery method.
Upgrade existing packages through the guarded flow	Plans requiring new packages or removals may be blocked.

Image/configuration rollback for containers belongs to Docker Manager. Package rollback does not rewind NAS data, application databases, container volumes or all system configuration.

NovaNAS application releases

The current update feed covers OS packages and managed tools. A packaged release pipeline for NovaNAS's own frontend/backend is not yet configured. Editing application source or rebuilding an ISO does not publish a new application package to other machines.

Networking and notifications

Keep management access reachable and route external traffic intentionally.

LAN access

First boot reports the hostname and discovered LAN addresses. mDNS advertises the `.local` hostname where the client network supports it. If name resolution fails, try the displayed IP address and confirm the client is on a reachable network.

For a stable address, use an intentional network configuration or DHCP reservation. Before changing bridges, addressing or routing, record the existing settings and keep a local console or independent recovery path available.

Domains and reverse proxies

The installed web server listens on port 80 and serves the frontend. For domain access, terminate HTTPS at a properly configured proxy and forward the required API and WebSocket routes. Do not publicly expose internal loopback API ports as a substitute for configuring the web entry point.

- Preserve the visible host, protocol and port through the proxy.
- Support WebSocket upgrade for Terminal and other long-lived connections.
- Route public `/s/` links and authenticated file-link management correctly.
- Test public access from outside the LAN, not only through local DNS.

Notifications

Notification Settings includes general options, email and Home Assistant configuration, test actions and save/reload feedback. Configure only the delivery services you use. Save the settings, then send a test and verify receipt at the destination.

The notification bell groups unread notifications, alerts and update information. A red bell indicates items needing attention; reading or clearing items is different from fixing the underlying system condition.

Credentials and diagnostics

SMTP credentials, Home Assistant tokens, camera URLs and cloud authorization data are sensitive. Redact them before sharing logs, configuration exports or screenshots. Do not embed them in documentation or installation images.

Security, backups and recovery

Protect data independently of the management interface and automatic rollback.

Access model

The installed backend runs as a restricted novanas system account. The installer-selected administrator authenticates separately; helpers and group membership grant the actions needed for storage, containers and virtualization. These capabilities are powerful and should not be treated as a tenant-isolation boundary.

- Use a strong administrator password and restrict management exposure.
- Protect API sessions, SSH keys, camera credentials and cloud account tokens.
- Use HTTPS for remote management and shared links when leaving a trusted LAN.
- Do not grant untrusted users Docker, libvirt or administrative helper access.

What to back up

Data class	Backup approach
Shares and application files	An independent copy with an explicit retention schedule and restore test.
Databases and container volumes	Application-aware exports or coordinated backups compatible with the running version.
VMs	Guest-aware backups or consistent stopped-guest disk copies, plus required VM configuration.
System configuration and secrets	Restricted, encrypted copies of relevant configuration; handle credentials separately.
Project sources and releases	Version control or external copies of sources, build configuration, ISO and validation reports.

Restore order

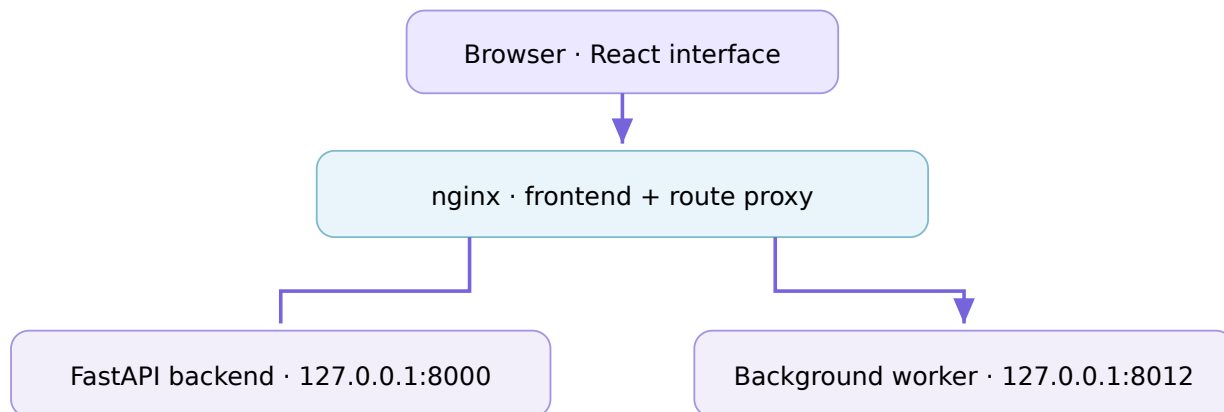
Identify the scope of the failure first. Restore OS and network access if needed, then storage visibility, permissions and configuration. Restore application data with a compatible application version. Start workloads in dependency order and verify real reads, writes and client access.

Test the restore, not just the backup

Keep at least one copy outside the failure domain of the NAS. A successful snapshot, archive creation or rollback message is not an end-to-end disaster-recovery test.

System architecture

The installed web server separates the static interface from API and background workers.



Constrained helpers · systemd · Docker · libvirt · storage · recording

Component	Responsibility
React / Vite frontend	Administrative workspaces, dialogs, charts and client-side flows. The installed system serves a production build.
FastAPI / Uvicorn backend	Authentication and APIs, with separate modules for storage, networking, surveillance, containers and VMs.
Independent worker on 8012	Protected Docker updates, supported VM clone/start routes and authenticated/public file-sharing routes.
systemd services and timers	Service lifecycle, monitoring, snapshots, cleanup and recovery work.
Linux storage and runtime tools	mdadm, Btrfs, Samba, Docker/Compose, QEMU/libvirt and controlled administrative helpers.

The development NAS can use Vite on port 5173; that development arrangement is different from the installer's nginx-served production frontend. Do not assume the development service layout and the installed service layout are identical.

Configuration and source map

Use the application sources and deployment templates as the authority for implementation details.

Location	Purpose
<code>/opt/nas/frontend/src</code>	React components and CSS. <code>App.jsx</code> hosts shared navigation and several workspaces.
<code>/opt/nas/frontend/public</code>	Manifest, web-app icons and other public frontend assets.
<code>/opt/nas/frontend/dist</code>	Built frontend served by the installed web server.
<code>/opt/nas/backend</code>	Backend Python modules and some application-specific runtime configuration.
<code>/opt/nas/backend/.venv</code>	Python runtime environment for backend services.
<code>/etc/novanas</code>	Administrator/authentication configuration and notification secrets; restrict access.
<code>/var/lib/novanas</code>	Persistent operational state such as notifications and update jobs.
<code>/usr/local/sbin</code>	Installed constrained NovaNAS helpers and maintenance controls.
<code>/usr/local/lib/novanas-updates</code>	Package rollback guard and supporting code.
<code>/var/lib/libvirt</code>	Libvirt-managed guest resources; actual disks can also use configured storage.

Important backend modules

`main.py` assembles the main application. Feature modules include `file_manager.py`, `novanas_terminal.py`, `novanas_network.py`, `novanas_docker_recovery.py`, `novanas_vm.py`, `novanas_vm_clone.py`, `novanas_surveillance.py`, `novanas_drive.py` and `novanas_recording_identity.py`.

Deployment templates

The separate `novanas-iso` project contains `config/` `service/proxy/` `first-boot` templates, `scripts/build-inside.sh`, `scripts/validate-image.py` and release artifacts. Refer to those templates when documenting a fresh installation; live host files may contain installation-specific identities and paths.

Do not copy runtime state into a release

Build from source and controlled defaults. Exclude credentials, sessions, recordings, personal homes, share databases, recovery journals and per-filesystem recording identity records.

Development and release workflow

Validate changes locally, then build and audit the deliverable you intend to distribute.

Frontend changes

Work on the active frontend sources, preserving shared components and theme variables. Verify the relevant user workflow in a browser; a successful build alone does not establish that a dialog, navigation transition or file operation works.

```
cd /opt/nas/frontend
npm run build -- --configLoader runner
```

The explicit config loader is the working build invocation for this project environment. Preserve the lockfile and declared dependencies when reproducing the build.

Backend changes

Check Python syntax and run focused tests with disposable fixtures. Backend source edits normally require the relevant process to reload before the live API changes. Plan restarts from an independent session; verify the actual endpoint and behavior afterward. Never use real recordings or user data as destructive test fixtures.

Installer build and repack

1. Use the separate ISO project; its full build runs through `./build-iso.sh` using Docker and Debian live-build.
2. Stage source and the built frontend. Apply clean-install administrator, portable-path and metrics defaults.
3. Run `scripts/validate-image.py` to reject host identity/state and check required features.
4. Build the compressed filesystem and hybrid ISO, preserving installer and boot configuration.
5. Extract the finished image, compare packaged files with staging, verify checksums and inspect BIOS/UEFI metadata.
6. Write the validation report. Delete an older release only after the replacement passes the required checks.

Application publishing is a separate project

A future release pipeline should create versioned packages, sign repository metadata, preserve configuration, define compatible migrations and health checks, and test rollback. The current ISO workflow does not automatically publish application releases to installed systems.

Operational checks and troubleshooting

Collect targeted evidence before changing service state or bypassing a safeguard.

Read-only checks

```
systemctl --failed
systemctl status novanas-backend.service --no-pager
journalctl -u novanas-backend.service -n 100 --no-pager
curl --fail http://127.0.0.1:8000/api/health
sudo dpkg --audit
```

Use the appropriate privileges and confirm unit names on the installed system. Some diagnostics can contain private paths, URLs or tokens; redact output before sharing it.

Symptom	Investigation
Web UI unavailable	Confirm address/DNS, nginx and backend state, network reachability and local health endpoint.
Terminal code 1006	Check WebSocket proxy headers/timeouts, detached session state and whether a backend restart occurred.
Update failed or interrupted	Read Installation Log, job status and recovery messages. Do not delete the recovery journal to clear the message.
Package kept back	Inspect the proposed dependency changes and the guarded updater's restrictions. Do not force a full upgrade without reviewing the plan.
Container update failed	Inspect the Docker Manager job, health-check result and rollback state; verify application data compatibility.
Retention identity mismatch	Verify filesystem/subvolume UUIDs and inodes. Investigate mount/storage changes; do not disable identity checks.
New sharing link fails	Confirm the browser origin, recipient reachability, public route, password, expiry and download limit.
Old or different web-app icon	Reload; if needed reinstall the cached web app. Android can apply its own launcher mask.
Live camera playback stalls	Check active encoder, fresh stream output, camera reachability and browser playback errors.

Distinguish a configuration check, a passing build and a running process from a successful end-to-end test. Confirm the action that matters to the user: a real download, healthy application, playable recording or booted guest.

Release baseline and verification

What is included, what was checked, and what remains unverified.

Current installer artifact

```
novanas-installer-amd64-20261030-185302.iso
```

Approximately 1.7 GB. SHA-256:

```
c65abfb439090814d626604a33c8b46ee8b367eb627738b6c287945fd07c9e28
```

Included changes

- Combined repository refresh and available-update check.
- Light-theme headers and opaque dialog surfaces.
- Rounded desktop/Apple web-app icons and matching Android artwork.
- Persistent Btrfs recording identity checks and safe retention integration.
- Terminal reconnect reuse for disconnected account-owned shells.
- Codex 0.166.0 and Antigravity 1.4.11, with new per-install account connection.
- Earlier domain-based file sharing, offline VM cloning, Docker/package rollback, weekly checks and installer fixes.

Verification scope

The published image passed ISO and squashfs checksum checks, internal filesystem SHA256/MD5 checks, packaged file comparison against audited staging, privacy checks, bundled CLI package/version checks and BIOS/UEFI metadata checks. Its adjacent validation report records the exact checks and prior feature test scope.

Not established by this release audit

This exact ISO was not boot-tested or fresh-installed. Secure Boot and hardware compatibility are not certified. Existing feature tests do not establish application-consistent data recovery or a complete disaster-recovery process.

Primary project references

- novanas-iso/README.md and PENDING-ISO-CHANGES.md.
- The artifact's .validation.md, .validation.json and .iso.sha256.
- novanas-iso/config/ and scripts/validate-image.py.
- Current frontend, backend, worker and update-guard sources listed in this guide.

Documentation is dated 30 October 2026. Recheck sources and validation reports when features or deployment templates change. Personal workstation applications and Home Assistant automations are outside the NovaNAS release.