anoBytes | TECHNOLOGY SOLUTIONS

Establish a secure server baseline on Linux

A new server setup blueprint: hostname and time, a stable address, an admin account, automatic updates, logging, and the disk and bootloader decisions, then CIS hardening with OpenSCAP. For Debian, Ubuntu Server, and RHEL.

SERVER BASELINE · DEBIAN / UBUNTU / RHEL · CIS, NIST, ISO · HOW-TO GUIDE

General educational guidance, not affiliated with any vendor or standards body named here. Test on a non-production system first. Published 2026-08-07.

11 steps, new server to a CIS scan

set up the server, harden it, then scan it against a CIS profile

5 releases covered

Debian 12 and 13 · Ubuntu Server 24.04 LTS · RHEL 9 and 10

A server baseline is the setup every new server gets first: hostname, clock, address, an admin account, patching, and logging. On a fresh build this takes minutes; retrofitting a live server takes much longer. This guide walks through that setup, hands the heavier hardening to a CIS Benchmark through OpenSCAP, and notes the CIS, NIST SP 800-53, and ISO/IEC 27001 controls each step supports, so the same work serves both engineering and audit.

SCOPE What this guide covers

The setup a CIS scan assumes, and the decisions it cannot make

OpenSCAP applies a CIS Benchmark better than any prose walkthrough, and the firewall has its own anoBytes guide, so this guide repeats neither: Steps 9 and 10 hand off to them. This guide covers what automation skips: hostname and time, the address, the admin account, automatic updates, logging, the disk and bootloader decisions a scan reports but cannot make for you, and which NIST and ISO controls each step covers.

Which releases, and why Ubuntu 24.04

Debian 12 and 13, Ubuntu Server 24.04 LTS, and RHEL 9 and 10, all systemd-based, so `hostnamectl`, `timedatectl`, `localectl`, and `systemctl` behave the same except where noted. Ubuntu is baselined on 24.04 because that is the release with current CIS and OpenSCAP content; 26.04 has none yet (the OpenSCAP guide covers this). Fedora users can follow the RHEL commands, then use the focused guides, which cover Fedora 44.

```
1. Foundation (this guide): identity, network, time, admin account, updates,
   logging + audit, access control, disk + bootloader
      |
2. Focused hardening (Step 9): host firewall (our guide) | SSH | sysctl | DNS
      |
3. Broad hardening (Step 10): a CIS profile via OpenSCAP (our guide)
      |
4. Verify and maintain (Step 11): re-scan on a schedule, watch for drift
```

Do the foundation on a fresh build before exposing the server. Steps 9 and 10 can run in either order, but running the CIS scan last shows what the focused hardening already fixed.

PREP Prerequisites and conventions

- A fresh or newly provisioned server you can still rebuild, and console or provider-recovery access in case a change goes wrong.
- Root access via `sudo` (or a root shell on a first boot that has no other account yet).
- Replace the placeholders: `host.example.com` is your server's fully qualified name, `alice` is your administrator's username, and `Region/City` is a timezone name from `timedatectl list-timezones`.
- On RHEL, register the system first (`sudo subscription-manager register`) or `dnf` has no repositories to install from. Red Hat's no-cost Developer Subscription for Individuals covers 16 systems.
- Package commands are shown per family: `apt` for Debian and Ubuntu, `dnf` for RHEL.

01 Identity: hostname, network, timezone, locale

Set a fully qualified hostname and map it locally, so the machine knows its own name before logging, certificates, or mail depend on it.

```
sudo hostnamectl set-hostname host.example.com
sudo timedatectl set-timezone Region/City      # e.g. America/New_York
```

Servers usually run UTC (`timedatectl set-timezone UTC`); set a local timezone only when something on the box needs local time.

Give the server a stable address. Two good ways: a static IP address configured on the host, or a DHCP reservation on your router or DHCP server, which keeps every address in one place and needs nothing on the host. For the host-side route the tooling differs by family. Find the interface name first with `ip -br link`, and on RHEL the connection name with `nmcli connection show`.

- **Change addresses from the console when you can**

  A wrong address, gateway, or interface name drops your SSH session with no way back in. Ubuntu's `netplan try` reverts on its own unless you confirm within two minutes; the nmcli and ifupdown paths have no such rollback, so run them from the provider console or with physical access.

```
# Ubuntu Server 24.04: netplan. Edit the YAML in /etc/netplan/ (root-owned, chmod 600):
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: false
      addresses: [192.0.2.10/24]
      routes:
        - to: default
          via: 192.0.2.1
      nameservers:
        addresses: [192.0.2.53]
# apply with automatic rollback unless you confirm:
sudo netplan try

# RHEL 9/10: NetworkManager (nmtui is the menu form of the same change)
sudo nmcli connection modify "<name>" ipv4.method manual ipv4.addresses 192.0.2.10/24 \
  ipv4.gateway 192.0.2.1 ipv4.dns 192.0.2.53
sudo nmcli connection up "<name>"

# Debian 12/13: /etc/network/interfaces, applied with ifdown/ifup from the console
auto eth0
iface eth0 inet static
    address 192.0.2.10/24
    gateway 192.0.2.1
```

On Debian, put DNS in `/etc/resolv.conf` (`nameserver 192.0.2.53`) unless a resolver manages that file for you.

- **On a cloud VM, set the address at the platform**

  Cloud instances get their network configuration from cloud-init, and it owns that config unless you disable its network handling (a drop-in in `/etc/cloud/cloud.cfg.d/` containing `network: {config: disabled}`). The better pattern is to leave the host on DHCP and pin the address where the platform manages it, the way an Elastic IP works in our nginx guide.

Add the name to `/etc/hosts`. Debian and Ubuntu map the hostname to `127.0.1.1`, a convention Debian documents for machines without a fixed address; on RHEL, map the machine's real IP:

```
# Debian / Ubuntu
127.0.1.1   host.example.com host
# RHEL (use the actual IP)
192.0.2.10  host.example.com host
```

Locale is the one identity setting that differs by family. RHEL ships locales in language packs; Debian generates them from `/etc/locale.gen`; Ubuntu's `locale-gen` also accepts the locale name directly:

```
# RHEL 9 / 10
sudo dnf install glibc-langpack-en
sudo localectl set-locale LANG=en_US.UTF-8

# Debian 12 / 13
sudo sed -i 's/^# en_US.UTF-8/en_US.UTF-8/' /etc/locale.gen
sudo locale-gen && sudo update-locale LANG=en_US.UTF-8

# Ubuntu 24.04
sudo locale-gen en_US.UTF-8 && sudo update-locale LANG=en_US.UTF-8
```

Verify with `hostnamectl status`, `hostname -f`, and `localectl status`. No benchmark rule requires any of this; the benchmarks only audit changes to files such as `/etc/hostname` (a Step 10 rule).

02 Time synchronization

Log correlation, certificate validation, and authentication all depend on an accurate clock. Each family already ships a time daemon, so confirm the right one is running rather than installing a second: Debian and Ubuntu use `systemd-timesyncd`, RHEL uses `chrony`.

```
timedatectl status                    # want: "System clock synchronized: yes"

# Debian / Ubuntu (systemd-timesyncd)
sudo systemctl enable --now systemd-timesyncd
timedatectl show-timesync --all

# RHEL (chrony)
sudo systemctl enable --now chronyd
chronyc sources -v
```

Run only one implementation. On Debian and Ubuntu, installing `chrony` removes the `systemd-timesyncd` package automatically; just do not add a second daemon by hand. Point either daemon at your organization's time source where one exists. Ubuntu is switching its default to chrony from 25.10, so expect chrony rather than timesyncd on releases after 24.04 LTS.

03 Administrative account and sudo

Create a named, non-root account for administration and grant it sudo. The group that carries sudo rights differs: Debian and Ubuntu use `sudo`, RHEL uses `wheel`, whose rule is already active in `/etc/sudoers`.

```
# Debian / Ubuntu
sudo adduser alice
sudo usermod -aG sudo alice

# RHEL (its adduser is a non-interactive alias of useradd)
sudo useradd alice && sudo passwd alice
sudo usermod -aG wheel alice
```

Confirm with `id alice` and `sudo -lU alice`. On a remote box, install the administrator's SSH key now (`ssh-copy-id alice@server` from your workstation), then log in as the new account over that path and repeat the sudo check before you rely on it. Disabling root's own SSH login belongs to your SSH hardening, once this account works. Password quality and lockout policy (PAM's `pwquality` and `faillock`, the login-stack modules that enforce them) arrive with the CIS profile in Step 10.

04 Automatic security updates

Unpatched packages are a common way into a server, so apply security updates without waiting for a human. Bring the system current first, then set up the automation:

```
# Debian / Ubuntu
sudo apt update && sudo apt full-upgrade
# RHEL
sudo dnf upgrade
```

Reboot if the upgrade brought a kernel. Debian and Ubuntu leave `/var/run/reboot-required` when one is needed, and Ubuntu Server's preinstalled needrestart also lists services still running old libraries after upgrades (on Debian, install it); on RHEL, compare `uname -r` with the newest installed `rpm -q kernel`.

Debian 12/13 and Ubuntu 24.04 use `unattended-upgrades`. Ubuntu Server enables it by default; on Debian, install and confirm it, since a Debian system may not have the package installed or enabled at all:

```
sudo apt install unattended-upgrades
sudo dpkg-reconfigure -plow unattended-upgrades   # answer Yes; writes 20auto-upgrades
sudo unattended-upgrade --dry-run -d              # show what a run would do
```

The default policy in `/etc/apt/apt.conf.d/50unattended-upgrades` takes updates from the security origin; the general `-updates` line ships commented out. `20auto-upgrades` is what enables the daily unattended run.

RHEL 9/10 use `dnf-automatic`, which is not installed by default. Set security-only apply in its config, then enable the one timer that reads that config:

```
sudo dnf install dnf-automatic

# /etc/dnf/automatic.conf
[commands]
upgrade_type = security
apply_updates = yes

sudo systemctl enable --now dnf-automatic.timer
```

- **Enable only dnf-automatic.timer**
  The package also ships preset timers (`dnf-automatic-install.timer` , `dnf-automatic-download.timer` , `dnf-automatic-notifyonly.timer`) that override the download and apply behavior in `automatic.conf` with fixed presets. Enable one of those alongside the main timer and the config file stops describing what actually happens. On CentOS Stream, `upgrade_type = security` matches nothing, because Stream publishes no security errata metadata; use `upgrade_type = default` there and accept full updates.

Our policy templates assume routine security updates install automatically. Schedule reboots for kernel updates, because the running kernel keeps its old code until the machine restarts.

05 Minimize installed software and services

Every listening service is attack surface. Inventory what runs, then disable or remove what the server does not need:

```
systemctl list-unit-files --type=service --state=enabled
sudo ss -tulpn                          # trace each listening port to its process

sudo systemctl disable --now <unit>     # stop now and at boot
# Debian / Ubuntu
sudo apt purge <pkg> && sudo apt autoremove --purge
# RHEL
sudo dnf remove <pkg> && sudo dnf autoremove
```

A service bound only to `127.0.0.1` is unreachable from the network; one bound to `0.0.0.0` or `[::]` is exposed if the firewall allows it. The CIS profile in Step 10 flags many removable services by name; clearing the obvious ones first makes that scan shorter to read.

06 Persistent logging and auditing

There are two log layers: the systemd journal for general logs, and the kernel audit trail for security events. Make both persist across reboots.

Make the journal persistent. `journald` stores logs on disk once `/var/log/journal/` exists. Debian 12/13 and Ubuntu 24.04 arrange this on a standard install, and Debian no longer installs `rsyslog` by default, so the journal is the primary log store there. Setting `Storage=persistent` explicitly guarantees it even on a cloud image that left the directory out:

```
sudo mkdir -p /etc/systemd/journald.conf.d
printf '[Journal]\nStorage=persistent\n' | \
  sudo tee /etc/systemd/journald.conf.d/99-persistent.conf
sudo systemctl restart systemd-journald
```

Enable auditd. It records security-relevant events (logins, privilege use, changes to watched files) and is the daemon the CIS logging rules configure:

```
# RHEL: the audit package is preinstalled; make sure it runs
sudo systemctl enable --now auditd

# Debian / Ubuntu: install it first
sudo apt install auditd audispd-plugins
sudo systemctl enable --now auditd
```

On RHEL, stop or restart auditd with `service auditd restart` rather than `systemctl` : the unit refuses a manual stop by design. Which events to record, where logs go, and how long to keep them are policy decisions; our Logging and Monitoring Policy template on the resources page is the companion for writing them down.

07 Mandatory access control

Keep the kernel's mandatory access control (MAC) enabled. It confines a compromised service to what its policy allows, a separate layer from file permissions or the firewall. Each family ships one by default; your job here is only to confirm nobody turned it off.

```
# RHEL 9/10: SELinux, enforcing with the targeted policy
getenforce            # want: Enforcing
sestatus

# Debian 12/13 and Ubuntu 24.04: AppArmor
sudo aa-status
```

Keep SELinux enforcing; use permissive only to debug a specific denial, and never disable it outright. When an application genuinely needs an exception, write a policy for that one case: the AL2023 SELinux boolean in our nginx reverse proxy guide is the pattern: one switch for one documented need.

08 Filesystem layout and the bootloader password

These two are build-time decisions. An OpenSCAP scan reports them as failures but cannot repartition a disk or invent a password for you, so decide them here.

Separate mounts with restrictive options. The CIS Benchmarks expect `/tmp`, `/dev/shm`, `/home`, `/var`, `/var/tmp`, `/var/log`, and `/var/log/audit` on their own mounts, carrying `nodev`, `nosuid`, and, where the mount's contents allow it, `noexec`. This is far easier at install time. On an existing single-volume cloud image, record the layout as an accepted risk in your baseline rather than repartitioning a live disk; the OpenSCAP guide treats those partition rules the same way.

Set a bootloader password so console access does not allow editing the kernel command line to bypass every control above. Leave UEFI Secure Boot on where the platform offers it: the password controls menu edits, Secure Boot protects the boot chain itself.

- **Decide the reboot behavior before you set it**

  With GRUB superusers configured, editing a menu entry always requires the password, and so does **booting** an entry unless that entry carries `--unrestricted`. On a server that must come back from an unattended reboot, make sure the default entry boots without a password: on Debian and Ubuntu add `--unrestricted` to the menu entries (via the `CLASS` line in `/etc/grub.d/10_linux`); RHEL's `grub2-setpassword` arranges this for you, requiring the password only to edit.

```
# RHEL 9/10 (writes /boot/grub2/user.cfg; edit-protection only)
sudo grub2-setpassword

# Debian / Ubuntu: generate the hash, then register it
grub-mkpasswd-pbkdf2 --iteration-count=600000 --salt=64   # copy the hash
# append to a custom /etc/grub.d file such as 40_custom:
#   set superusers="admin"
#   password_pbkdf2 admin grub.pbkdf2.sha512.600000.XXXX...
sudo update-grub
```

Store the bootloader password with your break-glass credentials, and test a reboot from the console before you need one.

09 Apply the focused hardening

Do the firewall first, with our dedicated guide, before the server faces a network you do not control. The other three are their own jobs:

Host firewall (our guide)

A default-deny inbound firewall with firewalld, ufw, or nftables, with safe SSH ordering so you do not lock yourself out. See the anoBytes guide *Configure and harden a host firewall on Linux* on our resources page. Its Ubuntu track applies to 24.04 as written. If the server will run containers, note that guide's warning: published container ports bypass the ufw and firewalld rules.

SSH hardening

Key-only authentication, root login disabled (the account from Step 3 is the way in), and a brute-force guard such as fail2ban. Do this before the server is exposed.

Kernel network hardening

The `sysctl` network parameters: reverse-path filtering, no redirects or source routing, SYN-flood protection. The CIS profile in Step 10 sets and checks these, which is one of the reasons to run it.

Encrypted DNS

Where the server's lookups should not be readable on the path, point `systemd-resolved` or your resolver at an encrypted upstream. Optional for most single-purpose servers; decide it deliberately.

- **If you run Debian**

  The firewall guide's distribution set is Fedora, RHEL, Ubuntu, and Arch-based systems, and Debian 12/13 track its Ubuntu commands: the same apt packaging and the same ufw front-end. One difference: Debian ships no firewall active by default and does not preinstall ufw, so run `sudo apt install ufw` first, then follow the guide's Track B, including its allow-SSH-first ordering.

10 Broad hardening: a CIS profile through OpenSCAP

With the foundation in place, measure and remediate against a CIS Benchmark using the anoBytes guide *Harden Linux against CIS Benchmarks with OpenSCAP*: scan, generate a fix for only what failed, review it, apply it, reboot, and re-scan. **Level 1** is the base recommendation, meant to be practical without an extensive performance or usability cost; start there on a general server. **Level 2** adds defense in depth for environments where security is paramount, and it can reduce functionality. The content available differs by distribution, so the hand-off is not identical:

DISTRIBUTION	OPENSCAP CIS CONTENT (v0.1.81)	HOW TO RUN IT
RHEL 9 / 10	`scap-security-guide` from AppStream (Red Hat-supported)	Exactly as the OpenSCAP guide describes
Ubuntu 24.04	Upstream release zip, `ssg-ubuntu2404-ds.xml`	As the OpenSCAP guide describes
Debian 12	Upstream release zip, `ssg-debian12-ds.xml`, profiles `cis_level1_server` / `cis_level2_server`	Same flow; install `openscap-scanner` from apt and swap the data stream file name

DISTRIBUTION	OPENSCAP CIS CONTENT (v0.1.81)	HOW TO RUN IT
Debian 13	**No CIS profile yet**; the `debian13` content carries ANSSI profiles plus a generic standard profile	See the note below

- **Two Debian caveats**
  The Debian 12 content in v0.1.81 aligns to the CIS Debian 12 Benchmark **v1.1.0**, while CIS has since published v2.0.0, so a clean scan shows alignment to the older set; say so to a client or an auditor. For Debian 13, until a content release ships a CIS profile, the accurate options are to apply the CIS Debian 13 Benchmark v1.0.0 by hand (or with CIS-CAT Pro, CIS's own assessor), or to scan with the ANSSI profiles the data stream does carry. ANSSI BP-028 is the hardening baseline of France's national cybersecurity agency, and a legitimate one; it is not CIS, so do not present one as the other.

ALIGN Where CIS, NIST, and ISO 27001 apply

Each step maps to a CIS Benchmark area and to the controls an auditor cites. CIS rule numbers change between versions, so the CIS column names the area and the Step 10 scan reports the exact rules. NIST and ISO titles are quoted from the standards.

BASELINE STEP	CIS BENCHMARK AREA	NIST SP 800-53 R5	ISO/IEC 27001:2022 ANNEX A
Identity and network (1)	n/a (site standard; no rule requires it)	CM-2 Baseline Configuration; CM-6 Configuration Settings	8.9 Configuration management
Time synchronization (2)	Services ▸ Time Synchronization	SC-45 System Time Synchronization	8.17 Clock synchronization
Admin account and sudo (3)	Access Control	AC-2 Account Management; AC-6 Least Privilege	8.2 Privileged access rights; 5.15 Access control
Automatic security updates (4)	Initial Setup ▸ Package Management	SI-2 Flaw Remediation	8.8 Management of technical vulnerabilities
Minimize software and services (5)	Services ▸ Server Services	CM-7 Least Functionality	8.19 Installation of software on operational systems
Logging and auditing (6)	Logging and Auditing	AU-2 Event Logging; AU-3 Content of Audit Records; AU-12 Audit Record Generation	8.15 Logging
Mandatory access control (7)	Initial Setup ▸ Mandatory Access Control	AC-3(3) Access Enforcement | Mandatory Access Control	8.3 Information access restriction

BASELINE STEP	CIS BENCHMARK AREA	NIST SP 800-53 R5	ISO/IEC 27001:2022 ANNEX A
Filesystem and bootloader (8)	Initial Setup ▸ Filesystem; Configure Bootloader	CM-6 Configuration Settings; IA-5 Authenticator Management	8.9 Configuration management; 8.5 Secure authentication

An ISMS also needs this baseline written down and owned. Our ISMS documentation structure and Shadow IT whitepapers and the policy templates, all on the resources page, cover that side.

11 Verify and maintain

A baseline drifts as the server changes, so keep checking it:

- Re-run the CIS scan on a schedule and after any significant change; a rising failure count is configuration drift.
- Keep the before-and-after OpenSCAP reports as evidence, as the OpenSCAP guide shows.
- Confirm the update timer is still active (`systemctl list-timers`), and that the journal and auditd still record after a reboot (`journalctl -b`, `sudo auditctl -s`).
- When the first real data lands, decide the backup story; the Business Continuity & Disaster Recovery Plan template on our resources page covers it.

EXTRAS Optional additions

Nothing here is required for the baseline. Take what fits the server.

A few everyday tools

Step 5 says keep the install small, and that stands. These are worth having on most servers: `htop` (process view), `tmux` (your session keeps running if the connection drops), `vim`, `curl`, `rsync`, `lsof`, `mtr` (path diagnostics), `ncdu` (what fills the disk), and `git`. All are in the standard repositories of Debian and Ubuntu; RHEL carries them all except `htop` and `ncdu`, which come from EPEL (Extra Packages for Enterprise Linux, the Fedora project's add-on repository). Install anything else a task calls for, and remove it once nothing needs it.

Lynis, a second opinion

Lynis (GPLv3, from CISOfy, which also sells an Enterprise version) is a one-script auditor with no daemon: `sudo lynis audit system` walks the host and prints warnings, suggestions, and a hardening index. It is a useful cross-check after the Step 10 scan and looks at things SCAP content does not. Its index is not CIS evidence, so keep the two reports separate. The distribution packages lag upstream; CISOfy's own package repository carries the current release.

rkhunter and chkrootkit

Readers ask about both. rkhunter's last release was in 2018; chkrootkit still gets releases (most recently January 2026); both remain signature-based spot checks. The CIS profile from Step 10 already brings AIDE, the file-integrity checker the benchmarks require (6.1.1 on RHEL, 6.3.1 on Debian and Ubuntu), which covers the same ground more durably. For detection beyond integrity checking, see the next block.

EDR, when you want live detection

auditd records events and AIDE compares files; neither watches an intrusion as it happens. That is EDR work. Two examples: Wazuh, open source and free to self-host, with an agent on each server reporting to a central manager, and Sandfly, commercial and agentless, which inspects hosts over SSH so nothing is installed on the endpoints. Either one is a project in its own right, so plan for that.

Tailscale for admin access

The same trade our nginx guide describes: a mesh VPN (WireGuard underneath) that takes SSH off the public internet, with Tailscale SSH controlling access through the tailnet policy. The costs are an agent on every host and a third party in the login path, and session recording needs a recorder node you run, on the plans that include it. Our WireGuard guide is the self-hosted version of the same idea.

Cloning this build into a template

Building once and cloning works as long as each clone gets its own identity. Reset two things in the template before snapshotting it. `/etc/machine-id` is the unique ID systemd generates at first boot; Ubuntu Server's netplan stack derives the DHCP client identifier from it, so clones sharing one can be handed the same address. Empty it with `sudo truncate -s 0 /etc/machine-id` and each clone writes its own at first boot. Clones that share SSH host keys can impersonate one another, so delete them in the template (`sudo rm /etc/ssh/ssh_host_*`). RHEL recreates missing keys when sshd starts; Debian and Ubuntu refuse to start sshd without them, so run `sudo ssh-keygen -A` on each clone from the console or a first-boot hook. Also remove `/var/lib/systemd/random-seed`, give each clone its own hostname and address (Step 1), and on RHEL register each clone rather than cloning the subscription. `virt-sysprep` (from the libguestfs tools) does all of this and more, offline, against a VM disk image; cloud platforms already do it per instance through cloud-init. If Wazuh or Tailscale went into the template, enroll each clone separately.

Ansible, for more than one server

This guide is a list of manual steps. If you run more than one server, doing them by hand on each machine leads to drift. Ansible is a common way to automate this, and it needs no agent: a managed node needs only SSH access and Python, so the account and key from Step 3 already prepare this server for it. The OpenSCAP guide can generate its CIS fixes as an Ansible playbook, so the Step 10 hardening can be applied the same way on every machine, and re-running the playbook on a schedule corrects the drift Step 11 checks for. A minimal install may not have `python3`, so install it where it is missing. And think before giving an automation account passwordless sudo: it is convenient, and it also means anyone holding that SSH key has root everywhere.

SHARE Optional: a shared group directory, with NFS to reach it

A team directory built from plain POSIX pieces, without Samba: a dedicated group, a setgid directory so new files inherit the group, and default ACLs so permissions hold as people write into it.

```
# the group and its members (members log in again before id shows it)
sudo groupadd projects
sudo usermod -aG projects alice

# the directory: group-owned, setgid, closed to others
sudo mkdir -p /srv/projects
sudo chgrp projects /srv/projects
sudo chmod 2770 /srv/projects
```

```
# default ACLs so new files stay group read-write
# (install the acl package first if setfacl is missing)
sudo setfacl -R -m g:projects:rwX -m d:g:projects:rwX /srv/projects
getfacl /srv/projects        # verify: a default:group:projects:rwx entry
```

The setgid bit (the 2 in 2770) makes files created inside inherit the `projects` group, and the default ACL (the `d:` entries) keeps them group-writable even when a member's umask says otherwise; an application that sets tighter permissions on its own files still wins. Nothing SELinux-specific is needed for local use.

Reaching it from other Linux machines: NFSv4

Plain NFS (`sec=sys`, the default) trusts the user and group IDs the client asserts; the NFS manual itself calls that adequate only between trusted hosts on a trusted network. So export to your own subnet or VPN range only, keep UIDs matched across machines, and treat Kerberos (`sec=krb5p`) as the authenticated upgrade, which needs a KDC and is beyond this guide.

```
# server: install and export (RHEL: sudo dnf install nfs-utils)
sudo apt install nfs-kernel-server
echo '/srv/projects 192.0.2.0/24(rw,no_subtree_check)' | sudo tee -a /etc/exports
sudo exportfs -ra

# firewall: an NFSv4 client needs only TCP 2049
sudo ufw allow from 192.0.2.0/24 to any port 2049 proto tcp
sudo firewall-cmd --permanent --add-service=nfs && sudo firewall-cmd --reload

# client: nfs-common (Debian/Ubuntu) or nfs-utils (RHEL), then mount
sudo mkdir -p /mnt/projects
sudo mount -t nfs -o nfsvers=4 192.0.2.10:/srv/projects /mnt/projects
# /etc/fstab:  192.0.2.10:/srv/projects  /mnt/projects  nfs  nfsvers=4  0 0
```

`root_squash` is on by default, so a client's root maps to an unprivileged user on the server; leave it that way. `no_subtree_check` is also the default; writing it in the export keeps `exportfs` from warning that no subtree option was named. On RHEL, SELinux allows the export as shipped (the `nfs_export_all_rw` boolean defaults on). The group membership and ACLs above do the authorization; NFS only carries the IDs, which is why the subnet scoping matters.

DONE What you achieved

- A repeatable server baseline: identity, a stable address, synchronized time, a named administrative account, automatic security updates, persistent logging with auditd, and mandatory access control enforcing.
- The build-time decisions made deliberately: the partition layout chosen or recorded as an accepted risk, and a bootloader password that does not block an unattended reboot.
- A clean hand-off to the focused firewall guide and to CIS hardening through OpenSCAP, with the Debian content caveats stated.
- Every step mapped to CIS, NIST SP 800-53, and ISO/IEC 27001, so one effort serves both engineering and audit.

REF References

CIS Benchmarks: Debian 12 v2.0.0, Debian 13 v1.0.0, Ubuntu 24.04 LTS v2.0.0, RHEL 9 v2.0.0, RHEL 10 v1.0.1 · `cisecurity.org/cis-benchmarks`

NIST SP 800-53 Rev 5, Security and Privacy Controls · `csrc.nist.gov` ISO/IEC 27001:2022 · `iso.org/standard/27001`

Ubuntu Server: automatic updates · `documentation.ubuntu.com/server` Debian: release notes and UnattendedUpgrades · `debian.org` · `wiki.debian.org/UnattendedUpgrades`

Red Hat: automating software updates with dnf-automatic; using SELinux; auditing the system · `docs.redhat.com`

ComplianceAsCode / content releases (the SCAP Security Guide) · `github.com/ComplianceAsCode/content` GNU GRUB manual (security) · `gnu.org/software/grub`