

Establish a secure server baseline on Linux

A new server setup blueprint: hostname and time, a stable address, an admin account, automatic security updates, the services to keep, logging and auditing, SELinux or AppArmor, and the disk, encryption and bootloader decisions. 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 2026-09-10.

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 13 and 12 · 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 you can show an auditor the work you already did.

SCOPE What this guide covers

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

OpenSCAP applies a CIS Benchmark rule by rule, which a prose walkthrough cannot do, 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 security updates, which services this server needs, logging and auditing, mandatory access control, and the disk, encryption and bootloader decisions a scan reports but cannot make for you.

Which releases, and why Ubuntu 24.04

Debian 13 and 12, 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 newest Ubuntu release with OpenSCAP CIS content: CIS publishes an Ubuntu 26.04 LTS Benchmark v1.0.0, but no data stream carries a CIS profile for 26.04 yet (the OpenSCAP guide covers this). Steps 1 to 9 still apply to 26.04, with one change: chrony replaced systemd-timesyncd there, so use the chrony commands in Step 2. Fedora users can follow the RHEL commands, then use the focused guides, which cover Fedora 44. Debian 12 has been on Debian LTS since 11 June 2026, so build a new server on Debian 13.

1. Foundation (this guide): identity, network, time, admin account, updates, services, logging + audit, access control, disk + encryption + bootloader
|
2. Focused hardening (Step 9): host firewall | SSH | DNS (our guides) | sysctl
|
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 itself, register the system first (`sudo subscription-manager register`) or `dnf` has no repositories to install from; the rebuilds and CentOS Stream need no registration. Red Hat's no-cost Developer Subscription for Individuals covers 16 physical or virtual nodes, for development and small production use, and is licensed to you personally rather than to your employer; check its terms before you rely on it for a production fleet.
- 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 server 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, though netplan's own manual asks you to check afterwards that the rollback really happened; 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 13/12: /etc/network/interfaces
auto eth0
iface eth0 inet static
  address 192.0.2.10/24
  gateway 192.0.2.1

# apply it from the console; there is no rollback if the address is wrong
sudo ifdown eth0; sudo ifup eth0
```

On Debian, put DNS in `/etc/resolv.conf` (`nameserver 192.0.2.53`) unless a resolver manages that file for you. Each example above sets IPv4. Where the network carries IPv6, set or record the IPv6 address the same way, because a server that autoconfigures one is reachable at an address nobody wrote down.

- **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's installer writes `127.0.1.1` for a machine with no permanent address, and Debian's own documentation says a machine that has one should use that address here instead. The server got one in the step above, so replace the installer's line rather than adding to it, and use the real IP on every family:

```
# Debian, Ubuntu, and RHEL alike (use the actual IP)
192.0.2.10  host.example.com host
```

Locale is the one identity setting that differs by family. RHEL packages 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 13 / 12
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`, which is a Level 2 auditd rule in Step 10.

02 Time synchronization

Log correlation, certificate validation, and authentication all need an accurate clock. Each family installs a Network Time Protocol (NTP) daemon by default, so confirm which one runs rather than adding 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.

The CIS Level 1 rules want a named time source, and the two families start from different places: Debian and Ubuntu install `timesyncd.conf` with every line commented out, so the rule fails until you write an `NTP=` line, while RHEL's `chrony.conf` carries a `pool` line that already satisfies it.

Ubuntu made chrony the default in 25.10, with Network Time Security (NTS, which authenticates the time source) enabled against the Ubuntu NTS pools, so 26.04 LTS installs chrony rather than timesyncd. chrony supports NTS on RHEL and Debian too, through `server <host> iburst nts` in `chrony.conf`, and SC-45(1) Synchronization with Authoritative Time Source asks for that.

If the machine starts with a badly wrong clock, NTS cannot validate the server's certificate and never syncs; `nocerttimecheck 1` in `chrony.conf` skips that check until the first clock update, and chrony warns to use it only where you have to, with a small set of trusted certificates.

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 server, 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.

On a provider image, list what is already there before you trust it (`awk -F: '$3>=1000' /etc/passwd` and `sudo find /home /root -name authorized_keys`): an image can come with an account or a key you did not create.

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 `/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 13/12 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 -pnow 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 and from the base stable archive; the general `-updates` line is commented out as installed. `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 installs three more timers (`dnf-automatic-install.timer`, `dnf-automatic-download.timer`, `dnf-automatic-notifyonly.timer`) whose units pass fixed flags that override the download and apply settings in `automatic.conf`. Enable one of those alongside the main timer and the config file stops describing what 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: set `reboot = when-needed` in `dnf-automatic`'s `[commands]` section, or `Unattended-Upgrade::Automatic-Reboot` with `Automatic-Reboot-Time` in `50unattended-upgrades`. On a server where an unplanned restart is expensive, download without installing (`download_updates = yes` with `apply_updates = no`, or `APT::Periodic::Download-Upgradeable-Packages "1"` with `APT::Periodic::Unattended-Upgrade "0"`) and apply in a window you control.

05 Minimize installed software and services

Every service that listens is another way in, so keep the set small. 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
```

Read what a removal proposes to take with it before you agree, since a package removal can pull dependants out with it and `autoremove` then widens the sweep. 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

Logging here has two layers, the systemd journal and the kernel audit trail, and both should persist across reboots.

Make the journal persistent. `journald` stores logs on disk once `/var/log/journal/` exists. Debian 13/12 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, and

the Level 1 rule wants the setting written down where journald is your logging system. RHEL 9 v3.0.0 dropped that rule when it consolidated on rsyslog, and the setting is still worth having there:

```
sudo mkdir -p /etc/systemd/journal.conf.d
printf '[Journal]\nStorage=persistent\n' | \
  sudo tee /etc/systemd/journal.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. Those rules sit at Level 2 in every benchmark this guide names, so a Level 1 scan will not prompt you for auditd at all; that is the reason it belongs here:

```
# RHEL: auditd is preinstalled and the distribution preset enables it; the plugins are not
systemctl is-enabled auditd
sudo dnf install audispd-plugins
sudo systemctl enable auditd && sudo service auditd start

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

audispd-plugins carries the audit dispatcher plugins and is the pair the CIS Debian and Ubuntu Benchmarks check for. On RHEL it is a separate package as well, and neither a default install nor the Step 10 remediation adds it, so without the line above `/etc/audit/plugins.d/` stays empty. On Ubuntu 24.04 it comes from universe, which Canonical patches only under Ubuntu Pro (free for personal use on up to five physical machines), so the automation in Step 4 will not.

On RHEL, start, stop, or restart auditd with **service** rather than **systemctl**: the unit refuses a manual stop by design, and Red Hat's guidance is that only **service** records the originating user ID correctly, leaving **systemctl** for **enable** and **status**. That command comes from **initscripts-service**, which the **audit** package recommends rather than requires, so it is present on an ordinary install and missing from an image built without weak dependencies.

Send a copy off the host. Root on this machine can delete the journal and the audit trail, so an attacker who reaches root removes the record of how they got in. Forward the logs to a collector this server's own administrator cannot edit. NIST SP 800-53 calls this AU-9 Protection of Audit Information, and AU-9(2) Store on Separate Physical Systems or Components asks for the off-host copy in those words.

CIS has two ways to do it. RHEL 9 v3.0.0, Debian 12 v2.0.0, and Ubuntu 24.04 v2.0.0 give you one route: journald forwards to rsyslog and rsyslog forwards to the collector, both Level 1, with TLS on that forwarding at Level 2. RHEL 10 v1.0.1 and Debian 13 v1.0.0 carry both routes at Level 1, so pick the one that fits and turn the other off. RHEL 10 audits that choice at rule 6.2.1.4, Ensure only one logging system is in use; Debian 13 has no matching rule and says in its section 6.1 that the recommendations are written so only one is implemented.

On RHEL 10 turning the other off means stopping rsyslog if you choose journal-upload, because rsyslog is installed and running by default and the scan's rule for that fails while both are active. The benchmark's own audit script is looser and passes as soon as either one is running, so the scan is the stricter of the two here.

The Step 10 content lags three of these editions and still carries both routes on RHEL 9, Debian 12 and Ubuntu 24.04, so a scan there holds journal-upload rules the current benchmark has dropped; the Step 10 callout gives the versions. Record which route you chose.

```
# journal-upload path (RHEL 10, Debian 13). Run systemd-journal-remote on the collector to receive.
sudo apt install systemd-journal-remote      # RHEL: sudo dnf install systemd-journal-remote
```

```
# /etc/systemd/journal-upload.conf
[Upload]
URL=https://logs.example.com:19532
ServerKeyFile=/etc/ssl/private/journal-upload.pem
ServerCertificateFile=/etc/ssl/certs/journal-upload.pem
TrustedCertificateFile=/etc/ssl/ca/trusted.pem

sudo systemctl enable --now systemd-journal-upload
```

The three certificate lines set up TLS in both directions. `ServerKeyFile` and `ServerCertificateFile` are this host's own key and certificate, which the collector checks; `TrustedCertificateFile` is the CA this host uses to confirm the collector is the one you meant. `systemd-journal-remote` checks the client against its own trusted CA unless it is started with `--trust=all`, and CIS asks for all four settings at Level 1.

On Debian and Ubuntu the systemd package already sets `ForwardToSyslog=yes` in `/usr/lib/systemd/journald.conf.d/syslog.conf`, and the name you give your own drop-in decides which of the two wins: systemd reads them from every directory in one filename order and applies the last, so anything beginning `99-` loses to `syslog.conf`. On this route write `ForwardToSyslog=no` into `/etc/systemd/journald.conf.d/zz-noforward.conf`, which sorts after it, and check with `systemd-analyze cat-config systemd/journald.conf`: the file printed last is the one in force. Stopping rsyslog leaves that setting where it is.

```
# rsyslog path (the only route on RHEL 9, Debian 12 and Ubuntu 24.04; also open to Debian 13).
# The collector has to accept TLS on 6514 and allow this host.
sudo dnf install rsyslog rsyslog-gnutls      # Debian / Ubuntu: sudo apt install rsyslog rsyslog-gnutls
printf '[Journal]\nForwardToSyslog=yes\n' | \
  sudo tee /etc/systemd/journald.conf.d/99-forward.conf

# /etc/rsyslog.d/40-forward.conf (the CA file is what verifies the collector)
global(DefaultNetstreamDriverCAFile="/path/to/loghost-ca.pem")
*. * action(type="omfwd" target="logs.example.com" port="6514" protocol="tcp"
  StreamDriver="gtls" StreamDriverMode="1"
  StreamDriverAuthMode="x509/name"
  StreamDriverPermittedPeers="logs.example.com"
  action.resumeRetryCount="100" queue.type="LinkedList" queue.size="1000")

sudo systemctl enable rsyslog
sudo systemctl restart systemd-journald rsyslog
```

What the Step 10 content checks on the journal-upload route, and what it reports whichever route you took, differs by release:

RELEASE	WHAT A LEVEL 1 SCAN REPORTS
RHEL 9	Checks the package and the socket. The URL, the certificates and the upload service are unimplemented, so the scan says nothing about them.
RHEL 10	Checks the package, the service and the receiving socket; the URL and the certificates are a manual recommendation, so the scan will not tell you whether TLS is set up. It fails the one-logging-system rule while rsyslog is still running, and fails <code>ForwardToSyslog</code> once you add the drop-in from the rsyslog block, since RHEL 10 installs that setting at <code>no</code> . Nothing credits the rsyslog side, so take journal-upload here for a clean Level 1 scan.
Debian 13	Checks the URL and the three certificate settings. The profile selects both <code>ForwardToSyslog=yes</code> and <code>ForwardToSyslog=no</code> at Level 1, so one of the two fails whichever route you take; note which rule that is.

RELEASE	WHAT A LEVEL 1 SCAN REPORTS
Debian 12	Checks the URL and the three certificate settings, and selects the same conflicting ForwardToSysLog pair as Debian 13.
Ubuntu 24.04	Checks the URL and the three certificate settings. Each rule is keyed on whether rsyslog is running, so only one of that pair applies.

Every distribution's content guards its journal-upload rules on whether rsyslog is disabled and whether **systemd-journal-remote** is installed, so on the rsyslog route they report as not applicable rather than as failures. RHEL 10 goes further and does not treat the two routes equally: it selects the journald rules and leaves the rsyslog routing rules unselected, with 6.2.3.4, the log file creation mode, the one rule it does select from the rsyslog subsection. No distribution's content checks whether rsyslog reaches a collector, because upstream marks that rule manual as site-specific.

Send the audit trail too. Both routes above carry journald's records. **auditd** keeps its own trail in **/var/log/audit/audit.log**, and **audispd-plugins** installs every dispatcher plugin with **active = no**, so nothing moves that trail until you turn one on:

- Setting **active = yes** in **/etc/audit/plugins.d/syslog.conf** puts the audit records into the journal, where the route you chose above carries them.
- Setting it in **/etc/audit/plugins.d/au-remote.conf** instead, with the collector's address in **/etc/audit/audisp-remote.conf**, sends them straight to an **auditd** on the collector, which needs **tcp_listen_port** set in its own **auditd.conf** to accept them.

Either way, restart with **sudo service auditd restart**. No CIS Linux rule asks for this, so the Step 10 scan will not prompt you.

Confirm the collector has the records before you count this step as done. **logger 'baseLine check'** puts a line in the journal and **sudo auditctl -m 'baseLine check'** puts one in the audit trail; find both on the collector. No CIS rule checks that the link works, so nothing in Step 10 will tell you if it stops.

The rsyslog block above is one-way TLS: the server proves its identity and the client does not. RHEL 9 v3.0.0 adds a Level 2 rule for the client's own certificate (6.1.2.13), so set **DefaultNetstreamDriverCertFile** and **DefaultNetstreamDriverKeyFile** there; Debian and Ubuntu stop at the CA file.

Either route sends everything the journal holds and keeps the local copy too, so size **/var/log**, the collector, and the link for that. 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 installs 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 13/12 and Ubuntu 24.04: AppArmor
sudo aa-status      # want: profiles loaded, the ones you rely on in enforce mode
```

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. Our nginx reverse proxy guide's AL2023 SELinux boolean is the pattern, one switch for one documented need.

If **getenforce** returns **Permissive** or **Disabled**, which some provider images have, turning SELinux back on takes more than an edit:

1. Set `SELINUX=permissive` in `/etc/selinux/config` and reboot.
2. Run `sudo fixfiles -F onboot` and reboot again to relabel the file system.
3. Move to `enforcing` once `ausearch -m AVC` comes back clean.

Red Hat sets out that order and warns to be in permissive mode for the relabel reboot, since unlabelled files that systemd needs can stop the machine booting. On Debian and Ubuntu, `sudo systemctl enable --now apparmor` loads the profiles again if `aa-status` shows none.

08 Filesystem layout, encryption at rest, and the bootloader password

These are build-time decisions. An OpenSCAP scan reports the partition layout and a missing bootloader password as failures, but it cannot repartition a disk or choose a password for you, and no CIS Linux rule covers encryption at rest at all.

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 Risk Register template on our resources page is where that record goes, and the OpenSCAP guide treats those partition rules the same way.

Decide encryption at rest. LUKS full-disk encryption is far easier to set up at install time; adding it to a running system means reencrypting the disk in place with a verified backup first, or a rebuild. It protects a disk that leaves your control, such as a decommissioned drive or a copied VM image, and does nothing against an attacker who reaches the running system. On a cloud instance the platform's volume encryption usually covers the same risk. An encrypted root also asks for a passphrase at every boot unless you set up an unattended unlock, so settle that too. Either way, record which one you chose.

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. `10_linux` is a packaged file, so a grub upgrade can offer to replace it: re-check the `CLASS` line after any grub update, and test a reboot.

```
mokutil --sb-state          # want: SecureBoot enabled (install the mokutil package if it is missing)

# 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. On a provider-hosted server that is reachable from the moment it boots, do this step as soon as Step 3 gives you a working account, then come back for Steps 4 to 8. The other three below 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. That guide is written for Ubuntu 26.04; its `ufw` commands apply to 24.04 unchanged. If the server will run containers, note that guide's warning: published container ports bypass the `ufw` and `firewalld` rules.

SSH hardening (our guide)

Key-only authentication, root login disabled (the account from Step 3 is the way in), and a brute-force guard such as `fail2ban`. See the anoBytes guide *Secure and harden SSH access on Linux* on our resources page. Do this before the server is exposed, or as your first move after Step 3 on a server that is already reachable.

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 (our guide)

Where the server's lookups should not be readable on the path, point `systemd-resolved` or your resolver at an encrypted upstream. See the anoBytes guide *Set up encrypted, validated DNS on Linux with Quad9* on our resources page. Optional for most single-purpose servers; decide it deliberately.

• If you run Debian or Ubuntu 24.04

All three guides above are written for Fedora 44, RHEL 9 and 10, Ubuntu 26.04, and Arch-based systems, so Debian 13/12 and Ubuntu 24.04 need a note:

- Debian 13/12 track the firewall guide's Ubuntu commands, the same `apt` packaging and `ufw` front-end, with one difference: Debian has no firewall active by default and does not preinstall `ufw`, so run `sudo apt install ufw` first, then follow that guide's Track B, including its allow-SSH-first ordering.
- Debian 13/12 and Ubuntu 24.04 follow the SSH guide's Ubuntu commands, since the packaging and the `sshd_config.d` layout are the same, with one gap: Debian 12 carries OpenSSH 9.2 and Ubuntu 24.04 carries 9.6, both older than the 9.8 that added `PerSourcePenalties`, so that guide's built-in rate limit is absent on those two and its Step 6 check finds nothing there; run `fail2ban` instead.
- For encrypted DNS, check which resolver the host runs: Ubuntu 24.04 installs `systemd-resolved`, Debian does not, so on Debian that guide's starting point is missing.

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	OPENS CAP CIS CONTENT	HOW TO RUN IT
RHEL 9 / 10	<code>scap-security-guide</code> from AppStream (Red Hat-supported)	Exactly as the OpenSCAP guide describes
Ubuntu 24.04	Upstream release zip, <code>ssg-ubuntu2404-ds.xml</code>	As the OpenSCAP guide describes
Debian 13	Upstream release zip, <code>ssg-debian13-ds.xml</code> , profiles <code>cis_level1_server</code> / <code>cis_level2_server</code>	Same flow; install <code>openscap-scanner</code> from apt and swap the data stream file name. The CIS profiles arrived in v0.1.82, so take that release or later
Debian 12	Upstream release zip, <code>ssg-debian12-ds.xml</code> , profiles <code>cis_level1_server</code> / <code>cis_level2_server</code>	Same flow as Debian 13

• **What the content aligns to, and where that lags CIS**

The Ubuntu 24.04 content aligns to the CIS Ubuntu 24.04 LTS Benchmark **v1.0.0** and the Debian 12 content to the CIS Debian 12 Benchmark **v1.1.0**. CIS has published v2.0.0 of both, so a clean scan on either shows alignment to the older set; say so to a client or an auditor. The RHEL 10 content tracks the current v1.0.1 and the Debian 13 content the current v1.0.0. The RHEL 9 content targets v2.0.0, which CIS superseded with v3.0.0 on 28 August 2026, so it lags in the same way; v3.0.0 cut the firewall section back to firewalld alone, renumbered SSH from 5.1.2 onward, and moved AIDE from 6.1.1 to 6.3.1.

• **Which content release you need**

On a content release older than v0.1.82 the `debian13` data stream carries no CIS profile, only ANSSI profiles and a generic standard profile. ANSSI BP-028 is the hardening baseline of France's national cybersecurity agency, and a legitimate one, but it is not CIS, so do not present one as the other.

The upstream figures above are v0.1.82. The AppStream build tracks its RHEL minor release instead, so check `rpm -q scap-security-guide` and the profile description before quoting an alignment. Upstream moved the RHEL 9 profiles to CIS v2.0.0 in SCAP Security Guide 0.1.74, which Red Hat released to RHEL 9 on 26 August 2024 as a batch update (`scap-security-guide-0.1.74-1.el9_4` , RHBA-2024:5811), so it arrived during 9.4 and every later minor inherited it. Red Hat rebases this package between minor releases as well as at them, so the version on the host is the one that counts; 9.8 carries 0.1.81, and a fully updated RHEL 9 has the v2.0.0 profiles.

ALIGN **Where CIS, NIST, and ISO 27001 apply**

Steps 1 to 8 and Step 11 each map to the controls an auditor cites, and to a CIS Benchmark area where one applies. 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

BASELINE STEP	CIS BENCHMARK AREA	NIST SP 800-53 R5	ISO/IEC 27001:2022 ANNEX A
Time synchronization (2)	Services ► Time Synchronization	SC-45 System Time Synchronization; SC-45(1) Synchronization with Authoritative Time Source	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-9 Protection of Audit Information; AU-9(2) Store on Separate Physical Systems or Components; 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
Filesystem, encryption, bootloader (8)	Initial Setup ► Filesystem; Configure Bootloader	CM-6 Configuration Settings; IA-5 Authenticator Management; SC-28 Protection of Information at Rest	8.9 Configuration management; 8.5 Secure authentication; 8.24 Use of cryptography
Verify and maintain (11)	n/a (re-run of the whole profile)	CA-7 Continuous Monitoring; CM-6 Configuration Settings	5.36 Compliance with policies, rules and standards for information security

An information security management system (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

Packages, configuration, and people change the server after the build, so re-check the baseline:

- 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`).
- Check that the collector is still receiving from this host, the way Step 6 does; the Step 10 scan does not test it.
- Watch the release's support horizon: past its security-update end date, the server stops receiving the patches Step 4 automates.
- Debian 12 left regular security support on 11 June 2026 and is on Debian LTS until 30 June 2028, on amd64, arm64, armhf, i386 and ppc64el only. Debian LTS is run by a separate group of volunteers and companies rather than the Debian security team, and it covers fewer packages; install `debian-security-support` and run `check-support-status` to see which of yours are outside it.
- Ubuntu 24.04 LTS has security updates for main until May 2029, and universe only under Ubuntu Pro.
- RHEL 9 has security errata until 31 May 2032 and RHEL 10 until 31 May 2035, through Red Hat's Full Support phase and then Maintenance Support, which keeps shipping them. Since 1 April 2025 Red Hat's standard errata criteria are Critical and Important CVEs plus Moderate ones scoring 7.0 or higher, so lower-scoring ones are left.
- When the first real data lands, decide how the server gets backed up; 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` , `lsuf` , `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 audit tool

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 (0.59, 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 10, 6.3.1 on RHEL 9 v3.0.0, Debian, and Ubuntu), which flags a change to any watched file rather than only known rootkit signatures. 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 what endpoint detection and response (EDR) does. 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 the machine ID and the SSH host keys 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. Writing `uninitialized` to it (`printf 'uninitialized\n' | sudo tee /etc/machine-id`) makes the next boot a first boot, and a first boot runs any unit with `ConditionFirstBoot=yes` . Emptying it instead (`sudo truncate -s 0 /etc/machine-id`) still gives each clone its own ID and leaves a file for a read-only image to bind-mount over, and systemd does not count that boot as the first, so those units do not run. systemd's manual accepts either.
- Clones that share SSH host keys can impersonate one another, so delete those in the template (`sudo rm /etc/ssh/ssh_host_*`). RHEL recreates missing keys when sshd starts. Debian 13 has an `sshd-keygen.service` that does the same on a first boot, though it carries `ConditionFirstBoot=yes` , so it runs only if you wrote `uninitialized` and truncating the file suppresses it; Debian 12 and Ubuntu 24.04 have no such unit and refuse to start sshd without keys. 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.

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 takes precedence. 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 physical 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 Kerberos key distribution center (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: serving an NFSv4 client needs only TCP 2049, and only from your subnet
sudo ufw allow from 192.0.2.0/24 to any port 2049 proto tcp
```

```
# firewalld: bind it to a source-restricted zone; --add-service=nfs alone
# opens 2049 to whatever the default zone accepts
sudo firewall-cmd --permanent --new-zone=nfsclients
sudo firewall-cmd --permanent --zone=nfsclients --add-source=192.0.2.0/24
sudo firewall-cmd --permanent --zone=nfsclients --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 blocks the export until you allow it: check with `getsebool nfs_export_all_rw` and turn it on with `sudo setsebool -P nfs_export_all_rw on` if it is off. It is off as installed, and it is host-wide rather than per-directory, since it lets the kernel's NFS server read and write every file type, so keep the export list itself narrow. The group membership and ACLs above do the authorization; NFS only carries the IDs.

Where both ends are new enough, RPC-with-TLS (RFC 9289) is far less work than Kerberos: `xprtsec=tl`s encrypts the transport, `xprtsec=mtls` also authenticates the client, and neither replaces `sec=sys` for user identity. It needs `tlshd` from `ktls-utils`:

- Debian 13, RHEL 9 and RHEL 10 carry it in their main repositories, and Red Hat documents NFS with TLS from RHEL 9.6.
- Ubuntu 24.04 has it in universe, so Canonical patches it only under Ubuntu Pro.
- Debian 12 has no `ktls-utils` package, so it cannot take either end of this.

DONE What you achieved

- A repeatable server baseline: identity, a stable address, synchronized time, a named administrative account, automatic security updates, only the services this server needs, persistent logging with auditd, and the kernel's mandatory access control left on (SELinux enforcing, or AppArmor loaded).
- The build-time decisions made deliberately: the partition layout chosen or recorded as an accepted risk, encryption at rest settled, and a bootloader password that does not block an unattended reboot.
- A clean hand-off to the focused guides and to CIS hardening through OpenSCAP, with the gaps in the RHEL 9, Ubuntu and Debian content stated.
- Steps 1 to 8 and Step 11 mapped to CIS, NIST SP 800-53, and ISO/IEC 27001, so you can show an auditor the work you already did.

REF References

CIS Benchmarks: Debian 12 v2.0.0, Debian 13 v1.0.0, Ubuntu 24.04 LTS v2.0.0, RHEL 9 v3.0.0 and v2.0.0, RHEL 10 v1.0.1 · cisecurity.org/cis-benchmarks NIST SP 800-53 Rev 5, Security and Privacy Controls for Information Systems and Organizations · csrc.nist.gov ISO/IEC 27001:2022 · iso.org

Ubuntu Server: automatic updates · documentation.ubuntu.com/server Debian: release notes and UnattendedUpgrades · debian.org · wiki.debian.org/UnattendedUpgrades ComplianceAsCode / content releases (the SCAP Security Guide) · github.com/ComplianceAsCode/content GNU GRUB manual (security) · gnu.org/software/grub

Red Hat: automating software updates with dnf-automatic; changing SELinux states and modes; auditing the system; managing file systems (NFS with TLS) · docs.redhat.com Life Cycle and errata policy · access.redhat.com/support/policy/updates/errata Advisory RHBA-2024:5811, scap-security-guide 0.1.74 · access.redhat.com/errata/RHBA-2024:5811 Red Hat Developer Subscription for Individuals FAQ · developers.redhat.com

Debian Long Term Support schedule · wiki.debian.org/LTS Debian Reference, chapter 5 (network setup, the 127.0.1.1 entry) · debian.org/doc/manuals/debian-reference Ubuntu release cycle · ubuntu.com/about/release-cycle Ubuntu Pro · ubuntu.com/pro debian-security-support · packages.debian.org/debian-security-support systemd manuals: [freedesktop.org/software/systemd/man](https://www.freedesktop.org/software/systemd/man) journald.conf(5) (drop-in precedence), systemd-analyze(1), journal-upload.conf(5), systemd-journal-upload(8), systemd-journal-remote(8), machine-id(5) · chrony-project.org/documentation.html chrony documentation, chrony.conf(5) (NTS and nocerttimecheck) · netplan.readthedocs.io Netplan reference, netplan-try(8) · github.com/linux-audit/audit-userspace auditctl(8), auditd.conf(5), audisp-remote(8), audisp-syslog(8) · github.com/linux-audit/audit-userspace

RFC 9289, Towards Remote Procedure Call Encryption by Default · rfc-editor.org/rfc/rfc9289 Linux nfs(5) and exports(5) manual pages · linux-nfs.org OpenSSH release notes (9.8, PerSourcePenalties) · openssh.com/releasenotes.html ANSSI BP-028, recommandations de configuration d'un système GNU/Linux · cyber.gouv.fr Lynis (CISOfy) · cisofy.com/lynis chkrootkit · chkrootkit.org Wazuh · wazuh.com Sandfly Security · sandflysecurity.com