SSH Hardening Reference

What This Page Covers

This page walks through every directive in `roles/ssh-baseline/templates/sshd_hardening.conf.j2` and explains how it lands on the target host. The deployed file is `/etc/ssh/sshd_config.d/10-pbr-hardening.conf`.

The hardening is aligned with CIS Ubuntu Linux 22.04 Benchmark v2.0.0. Where we deviate, it's documented inline and below.

How the Config Reaches sshd

Drop-in directory pattern

Ubuntu's `sshd_config` reads drop-in files from `/etc/ssh/sshd_config.d/` via an `Include` directive. Cloud-init images have this by default; some ISO installs do not. The role ensures the include is present:

```
- name: Ensure sshd_config has Include directive for drop-ins
  ansible.builtin.lineinfile:
    path: /etc/ssh/sshd_config
    line: "Include /etc/ssh/sshd_config.d/*.conf"
    insertbefore: BOF
    state: present
    validate: "/usr/sbin/sshd -t -f %s"
  notify: Restart sshd
```

Why insert at BOF (beginning of file): sshd uses first-match-wins semantics for most directives. Placing the Include directive at the top of `sshd_config` means drop-ins are evaluated first — our hardening directives win over any conflicting directive later in the base config.

Filename prefix: 10-

The deployed file is named `10-pbr-hardening.conf`. Drop-ins are loaded in lexicographic order. The `10-` prefix ensures our file loads before Ubuntu's default `50-cloud-init.conf`, which sets `PasswordAuthentication yes`. Without the `10-` prefix and first-match-wins, cloud-init's value could win.

Validation gating

Both the Include line and the hardening file are written with `validate: "/usr/sbin/sshd -t -f %s"`. Ansible writes to a temp file, runs `sshd -t -f <tempfile>` against it, and only moves the temp file into place if validation passes. After the file is in place, the role also runs a final `sshd -t` against the live combined config (defence in depth).

The Hardening File: Full Source

Template: `roles/ssh-baseline/templates/sshd_hardening.conf.j2`. Rendered output (all variables substituted with their defaults):

```
# PBR SSH Hardening - Managed by Ansible, do not edit manually
# CIS Ubuntu Linux 22.04 Benchmark v2.0.0 aligned

Port 22
LogLevel VERBOSE
LoginGraceTime 60

# === Authentication ===
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
KbdInteractiveAuthentication yes
AuthenticationMethods publickey,keyboard-interactive
MaxAuthTries 3
GSSAPIAuthentication no
UsePAM yes
UseDNS no
```

```
# === Compliance affirmations (defaults made explicit for audit evidence) ===
IgnoreRhosts yes
HostbasedAuthentication no
PermitEmptyPasswords no
PermitUserEnvironment no

# === Session management ===
MaxSessions 4
MaxStartups 10:30:60
ClientAliveInterval 300
ClientAliveCountMax 2

# === Forwarding ===
AllowTcpForwarding no
X11Forwarding no
AllowAgentForwarding no

# === Other hardening ===
Compression no
TCPKeepAlive no

# === Modern crypto ===
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-
ctr,aes192-ctr,aes128-ctr
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com
KexAlgorithms sntrup761x25519-sha512@openssh.com,curve25519-sha256,curve25519-
sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256

# === Legal banner ===
Banner /etc/issue.net

# === Access control ===
AllowGroups sudo sg_serveraccess sg_sudo

# === SSH key retrieval ===
AuthorizedKeysFile none
AuthorizedKeysCommand /usr/bin/sss_ssh_authorizedkeys %u
AuthorizedKeysCommandUser nobody

# === Break-glass: pbr_admin ===
```

```
Match User pbr_admin Address 10.0.0.0/8,192.168.0.0/16
    PasswordAuthentication yes
    AuthenticationMethods password

# === Ansible automation account ===
Match User ansible
    AuthorizedKeysFile .ssh/authorized_keys
    AuthenticationMethods publickey
    KbdInteractiveAuthentication no
```

Directive Walkthrough

Authentication block

Directive	Value	Notes
`PermitRootLogin`	`no`	Root never logs in directly. Use `pbr_admin` + sudo or AD user + sudo.
`PasswordAuthentication`	`no`	Disabled globally. Re-enabled only inside the `pbr_admin` Match block.
`PubkeyAuthentication`	`yes`	Required by all flows except `pbr_admin`.
`KbdInteractiveAuthentication`	`yes`	Required for Duo PAM keyboard-interactive challenge. Disabled in `ansible` Match block.
`AuthenticationMethods`	`publickey,keyboard-interactive`	Both required. Overridden per-user in Match blocks for `pbr_admin` (password) and `ansible` (publickey only).
`MaxAuthTries`	`3`	Per-connection auth attempt limit.
`GSSAPIAuthentication`	`no`	We don't use GSSAPI/Kerberos for SSH auth. AD password validation happens via PAM/SSSD, not via Kerberos ticket forwarding.
`UsePAM`	`yes`	Required — Duo and pam_sss live in PAM.

Directive	Value	Notes
`UseDNS`	`no`	Don't reverse-resolve client IPs into hostnames. Eliminates a slow DNS lookup on every connection and avoids confusion when client reverse-DNS is broken.

Compliance affirmations

These four directives are defaults in OpenSSH but stated explicitly for audit evidence:

Directive	Value	What it prevents
`IgnoreRhosts`	`yes`	.rhosts / .shosts trust files cannot be used for auth.
`HostbasedAuthentication`	`no`	Trust-by-host-key auth disabled.
`PermitEmptyPasswords`	`no`	Empty passwords cannot authenticate. (Belt-and-braces; `PasswordAuthentication no` already disallows.)
`PermitUserEnvironment`	`no`	Users cannot inject environment vars via `~/.ssh/environment` — prevents PATH/LD_PRELOAD-style attacks.

Session management

Directive	Value	Notes
`MaxSessions`	`4`	Concurrent multiplexed sessions per SSH connection. CIS recommendation.
`MaxStartups`	`10:30:60`	Up to 10 unauth'd connections; from 10-60, drop 30% randomly; reject at 60. Mitigates connection-exhaustion DoS.
`ClientAliveInterval`	`300`	Send keepalive probes every 5 minutes.
`ClientAliveCountMax`	`2`	Drop the connection after 2 missed keepalives. Idle sessions die after 10 minutes.

Forwarding (all disabled)

Directive	Value	What it prevents
`AllowTcpForwarding`	`no`	Local/remote port forwarding. No tunnel-the-DB-over-ssh patterns.
`X11Forwarding`	`no`	Graphical apps via X over SSH. Unused at PBR.
`AllowAgentForwarding`	`no`	Forwarding ssh-agent to the remote host (would let a malicious admin on the remote pivot using your keys).

Other hardening

Directive	Value	Notes
`Compression`	`no`	Compression has historically been a source of side-channel attacks (CRIME-style).
`TCPKeepAlive`	`no`	Use SSH-level keep-alive (ClientAliveInterval) instead. TCPKeepAlive is unauthenticated and spoofable.

Modern Crypto

Ciphers

```
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

- AEAD ciphers preferred (chacha20-poly1305, aes-gcm) — encryption and integrity combined.
- aes-ctr modes retained for client compatibility with older OpenSSH releases (paired with hmac-sha2 in MACs).
- CBC modes and legacy 3DES/RC4/Blowfish/arcfour are all excluded.

MACs

```
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com
```

- `-etm` (Encrypt-then-MAC) only. Authenticates the ciphertext, preventing oracle attacks on the plaintext.
- SHA-2 family or umac-128. SHA-1 MACs are excluded.

Key Exchange (with post-quantum hybrid)

```
KexAlgorithms sntrup761x25519-sha512@openssh.com,curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256
```

- `sntrup761x25519-sha512@openssh.com` — post-quantum hybrid KEX. Combines NTRU Prime (PQ) with X25519 (classical) so the resulting key is secure unless both are broken. Available in OpenSSH 9.0+.
- curve25519 fallbacks for clients without PQ support.
- ECDH (P-521, P-384, P-256) as classical fallbacks.
- SHA-1-based KEX, RSA-based KEX, and DH group 1/14 are all excluded.

Access Control: AllowGroups

```
AllowGroups sudo sg_serveraccess sg_sudo
```

sshd's `AllowGroups` is a hard allow-list checked early in the connection. A user must be in **at least one** listed group to even reach the authentication phase. Users not in any listed group get rejected with "User <user> from <ip> not allowed because none of user's groups are listed in AllowGroups".

The three groups:

Group	Origin	Members
`sudo`	Local Unix group	`ansible` (added by role preconditions), `pbr_admin` (added by manual bootstrap)
`sg_serveraccess`	AD group (SSSD-mapped)	AD users with SSH access (no sudo)
`sg_sudo`	AD group (SSSD-mapped)	AD users with sudo

Group names from AD are lowercased by SSSD when mapped to local POSIX groups, so the lowercase form is what sshd matches against.

Why include local `sudo` rather than special-casing `ansible` and `pbr_admin` via Match blocks: Match blocks override settings; they don't bypass `AllowGroups`. The user must qualify at the global level first. Listing `sudo` in `AllowGroups` is the simplest way to permit the two local

accounts.

v2.4.1 corollary: Because `AllowGroups sudo` is what permits the `ansible` account to connect, the role must ensure `ansible` is in the local `sudo` group before the hardening config takes effect. That's done idempotently in `preconditions.yml`.

Access Control: AuthorizedKeysCommand

```
AuthorizedKeysFile none
AuthorizedKeysCommand /usr/bin/sss_ssh_authorizedkeys %u
AuthorizedKeysCommandUser nobody
```

Three lines that change the default sshd key retrieval flow entirely:

- **`AuthorizedKeysFile none`** — disable the default file-based lookup (`~/.ssh/authorized_keys`). Critical: prevents AD users from bypassing AD-managed key revocation by writing their own key files.
- **`AuthorizedKeysCommand /usr/bin/sss_ssh_authorizedkeys %u`** — for each connection, sshd runs this command with the username, expects valid `authorized_keys`-format output on stdout.
- **`AuthorizedKeysCommandUser nobody`** — run the command as `nobody`. This is OpenSSH and SSSD's documented recommendation: the command should run as a low-privilege user.

The `sss_ssh_authorizedkeys` binary queries the SSSD ssh responder, which queries AD via LDAP for the user's `sshPublicKey` attribute. See **AD Integration & SSSD** for the full flow.

Match Block: pbr_admin (break-glass)

```
Match User pbr_admin Address 10.0.0.0/8,192.168.0.0/16
    PasswordAuthentication yes
    AuthenticationMethods password
```

Match conditions are AND-ed: the user must be `pbr_admin` AND connecting from one of the listed CIDRs. If both match, the block's directives override the global config for this connection only.

The overrides:

- `PasswordAuthentication yes` — re-enable password auth (globally `no`).
- `AuthenticationMethods password` — this user authenticates with password only (globally `publickey,keyboard-interactive`).

The source address list is templated from `pbr_admin_allowed_sources` in defaults. CIDR list, comma-separated, no spaces — per sshd_config(5) syntax.

Current scope (2026-06-17): widened to `10.0.0.0/8,192.168.0.0/16` — the break-glass account is reachable from anywhere on the internal 10/8 estate. As `pbr_admin` is password-only (carved out of the Duo PAM flow), this is a deliberately broad source scope; see the Version History on the *Known Limitations, Troubleshooting & Version History* page for the rationale and the plan to narrow it to per-site management CIDRs.

Important: this Match block does *not* bypass `AllowGroups`. `pbr_admin` must still be in `sudo` (handled by manual bootstrap, verified by preflight).

Match Block: ansible (automation)

```
Match User ansible
    AuthorizedKeysFile .ssh/authorized_keys
    AuthenticationMethods publickey
    KbdInteractiveAuthentication no
```

The `ansible` account is local-only and has no AD-side key. The overrides:

- **`AuthorizedKeysFile .ssh/authorized_keys`** — re-enable file-based key lookup (overrides global `none`). Bootstrap script installs the control node's public key here.
- **`AuthenticationMethods publickey`** — publickey is sufficient (overrides global `publickey,keyboard-interactive`). The ansible account skips PAM entirely on auth.
- **`KbdInteractiveAuthentication no`** — explicitly disable the keyboard-interactive flow for this user. Belt-and-braces with `AuthenticationMethods publickey`.

This is what lets Ansible run non-interactively, without Duo prompts, against every host.

Banner

```
Banner /etc/issue.net
```

The banner file is deployed by `roles/ssh-baseline/tasks/sshd.yml` from `roles/ssh-baseline/files/issue.net`. The banner displays before authentication — useful for legal notice and unauthorised-access deterrence.

Note: the banner content is in `files/issue.net` — not templated and not currently in the code dump. To inspect the deployed banner: `cat /etc/issue.net` on any baselined host.

Validation Flow

The role validates SSH config three times during deployment:

1. **During the Include directive write:** `lineinfile` validates via `sshd -t -f <tempfile>`. Catches a broken include line.
2. **During the hardening file write:** `template` validates via `sshd -t -f <tempfile>`. Catches a broken hardening directive before the file lands.
3. **After both files are in place:** `sshd -t` against the live combined config. Catches conflicts between the two files (which the per-file validation can't see).

Only after all three pass does the handler restart sshd.

Notes on Port 22 vs Custom Ports

From the inline comment in `defaults/main.yml`:

> ssh_port stays at 22. On Ubuntu 22.10+ and 24.04 LTS, OpenSSH uses systemd socket activation by default. If `ssh_port` is changed, `/etc/systemd/system/ssh.socket.d/` overrides must also be managed, or `ssh.socket` disabled in favour of `ssh.service`.

The role does not currently manage `ssh.socket` overrides. Changing `ssh_port` from 22 would require additional task work and is intentionally not supported until needed.

Where to Read Next

- **Configuration Reference** — the full list of SSH-related variables and how to override them
- **Duo MFA Integration** — the keyboard-interactive challenge that this hardening enables
- **AD Integration & SSSD** — how `sss_ssh_authorizedkeys` retrieves AD-stored keys

---

Revision #2
Created 2026-05-13 05:31:27 UTC by PBR_Documentation
Updated 2026-06-17 07:58:47 UTC by PBR_Documentation