TUTORIAL

How to Secure SSH on CentOS 7 with Key-Based Authentication

This tutorial shows how to move a CentOS 7 SSH server from password login to key-based authentication, validate the change safely, and confirm remote access before production.

Operating Systems - August 09, 2026

Introduction

The practical problem is simple: password-based SSH access on a CentOS 7 server is easy to brute force, easy to share, and hard to audit cleanly. If your server is still accepting passwords over SSH, you are leaving remote administrative access exposed to credential attacks that often succeed long before anyone notices unusual login attempts.

In this tutorial, you will secure SSH on CentOS 7 by introducing key-based authentication, testing it from a second session, and then disabling password login only after you have verified that key access works. The end state is a server that accepts SSH logins with approved private keys, rejects password authentication, and can be validated before you rely on it in production.

Before you change anything

Goal

Make sure you can complete the transition without locking yourself out of the server.

Stop-here-if warnings

Stop here if any of the following are true:

- You do not already have console access, out-of-band access, or another trusted recovery path.
- You are connected over SSH and do not have a second session available for testing.
- You do not know the current remote username, sudo access state, or whether SSH is restricted by firewall or network ACLs.
- You cannot afford to lose access if the SSH daemon is reloaded or if a configuration mistake is introduced.

Key-based SSH hardening is safe when you can validate changes in parallel with the current login method. It is risky when you cannot recover independently.

Action

Confirm the following on the target CentOS 7 host:

- SSH service is running.
- You have at least one administrative account that can be used for testing.
- Your local machine can generate and store an SSH key pair securely.
- Port 22, or your chosen SSH port, is reachable from your admin workstation.

Expected output

You should know which account will receive the public key, which source host will connect, and how you will recover if a mistake blocks access.

Validation

Check the SSH service status:

Confirm the configuration file exists:

Common failure

The most common failure at this stage is assuming you can ?just fix it later.? If you disable password login before you confirm a working key-based login from a separate session, you can lock yourself out.

Generate an SSH key pair on your client

Goal

Create a private/public key pair that will be used for authentication.

Action

On your local workstation, generate a modern key pair if you do not already have one:

If your environment does not support Ed25519 for a specific compatibility reason, use RSA with a strong key size and a passphrase. Keep the private key protected with file permissions and, where practical, a passphrase.

Expected output

You should have a private key stored locally and a matching public key that can be copied to the server.

Validation

List the created files:

Inspect the public key if needed:

Common failure

A frequent mistake is generating the key without a passphrase in a shared environment or saving the private key with overly broad permissions. If the private key is exposed, the security benefit is reduced significantly.

Install the public key on the CentOS 7 server

Goal

Place the public key into the target account?s authorized_keys file with the correct ownership and permissions.

Action

The simplest safe method is ssh-copy-id from your workstation:

If you need to do it manually, log in once with your current method and create the .ssh directory and authorized_keys file for the target account:

Ensure the target account owns the files:

If you are standardizing SSH hardening across multiple Linux hosts, the same validation pattern applies in broader hardening workflows such as Hardening CentOS SSH with Key-Based Authentication and Fail2ban.

Expected output

The server should contain the public key in the target user?s ~/.ssh/authorized_keys, and SSH should still allow the existing login method until you explicitly disable it later.

Validation

Check permissions and ownership:

A typical safe result is:

- ~/.ssh owned by the target user
- permissions set to 700
- authorized_keys owned by the target user
- permissions set to 600

Common failure

If SSH still rejects key login after installation, the usual causes are incorrect ownership, wrong permissions, the key pasted into the wrong account, or a malformed key line with extra spaces or line breaks.

Test key-based login before changing SSH policy

Goal

Verify that the server accepts the new key while password authentication still works as a fallback.

Action

Open a new terminal session from your workstation and try key-based login explicitly:

If your key is the default identity, the explicit -i option helps remove ambiguity during testing.

Expected output

You should authenticate without being prompted for the account password. If the private key has a passphrase, you may be prompted for that passphrase instead.

Validation

Run a non-interactive command after login to confirm the session is usable:

Expected output should show the server hostname and the correct remote username.

Common failure

If login falls back to password or fails completely, stop and investigate before changing sshd_config. Typical causes include:

- incorrect key installed for the account
- private key file not readable by the client user
- sshd configuration overriding key authentication
- SELinux or file context issues after manual file placement

Configure sshd to prefer keys and reject passwords

Goal

Change the SSH daemon so that approved keys are required for interactive authentication.

Action

Edit /etc/ssh/sshd_config carefully:

Set or confirm the following directives:

In many environments, you may also want to prevent empty passwords and direct root login:

Be cautious: changing PermitRootLogin can affect operational workflows if your maintenance process still depends on direct root SSH access. Confirm your administrative model before enforcing it.

Save the file, then validate the syntax before reloading SSH:

Expected output

No output from sshd -t indicates the syntax is acceptable.

Validation

Check the effective SSH configuration after reload or restart:

You want to confirm that the daemon is actually using the expected values, not just that the file contains them.

Common failure

A common mistake is editing the file but never verifying the effective configuration. Another is reloading sshd with a syntax error present; that can prevent the daemon from applying your changes correctly. Always run sshd -t before reload.

Lock down access only after a second login test

Goal

Ensure the server is still reachable by key before making password login unavailable.

Action

From a second terminal session, log out and log back in using the SSH key. Do not assume the first successful key login is enough. Verify you can start a fresh session after the config change.

Then test that password authentication is rejected. You can observe this by attempting login without specifying a key or by using a user that previously relied on password authentication.

Expected output

- Key-based login succeeds.
- Password-based login is rejected or no longer offered.
- Administrative access remains intact through the approved key path.

Validation

You can use a verbose SSH connection to see what method the client attempts:

Look for public key authentication being attempted and password authentication no longer succeeding.

Common failure

If password access is still possible, re-check for duplicate directives in sshd_config or included configuration fragments, and confirm you reloaded the correct service. If key login fails after disabling passwords, restore access from your backup config and re-enable password auth temporarily only long enough to recover.

Verify the server-side controls

Goal

Confirm the server is enforcing the intended authentication policy and not relying on assumptions.

Action

Inspect the live daemon settings:

Check that the key file exists and is readable only by the intended user:

If SELinux is enforcing on your CentOS 7 host, verify file contexts are not broken after manual edits:

Expected output

The daemon should report key authentication enabled and password authentication disabled. The .ssh directory and authorized_keys file should have strict permissions and correct ownership.

Validation

Review the SSH authentication path in logs after a login attempt:

Successful key logins should be visible in the audit trail, which helps confirm that the expected authentication method is being used.

Common failure

If logs show authentication errors but the syntax is valid, the issue is often file permissions, SELinux context, or an unexpected override in a secondary SSH configuration file.

Operational follow-up

Goal

Keep the server secure after the initial switch to key-based authentication.

Action

Adopt a few operational habits that reduce the chance of regression:

- Keep the private key protected with a passphrase where feasible.
- Remove stale keys from authorized_keys when staff or automation changes.
- Use a separate administrative account for SSH instead of direct root access.
- Re-test SSH access after configuration management changes or package updates.
- Consider pairing key-based SSH with rate limiting or brute-force mitigation; controls such as password removal and tools like Fail2ban complement each other, as described in Hardening CentOS SSH with Key-Based Authentication and Fail2ban.

Expected output

Your SSH access model should remain predictable: only approved keys can authenticate, changes are auditable, and recovery procedures are still available if a key is lost.

Validation

Perform a periodic access check from an admin workstation:

Confirm that unexpected accounts cannot log in with passwords and that the approved key still works.

Common failure

The most common long-term failure is configuration drift. A later change to sshd_config, a copied key file with weak permissions, or a forgotten emergency account can quietly undo the hardening you just completed.

What the finished state should look like

When this is implemented correctly, CentOS 7 SSH access should have a narrow, testable trust path:

- the target user has a valid public key in authorized_keys
- the private key remains on the client side
- SSH accepts key-based authentication
- password authentication is disabled for SSH
- the configuration has been syntax-checked and validated live
- you have confirmed access from a second session before relying on the change

That is the practical standard for a secure SSH deployment on CentOS 7: you can still administer the server remotely, but only through keys you control and can validate.

Use this guidance together with CentOS 8 SSH hardening to connect the workflow with related operational context already available on the site.

Use this guidance together with SELinux enforcing mode CentOS 7 and Ubuntu login activity audit to connect the workflow with related operational context already available on the site.