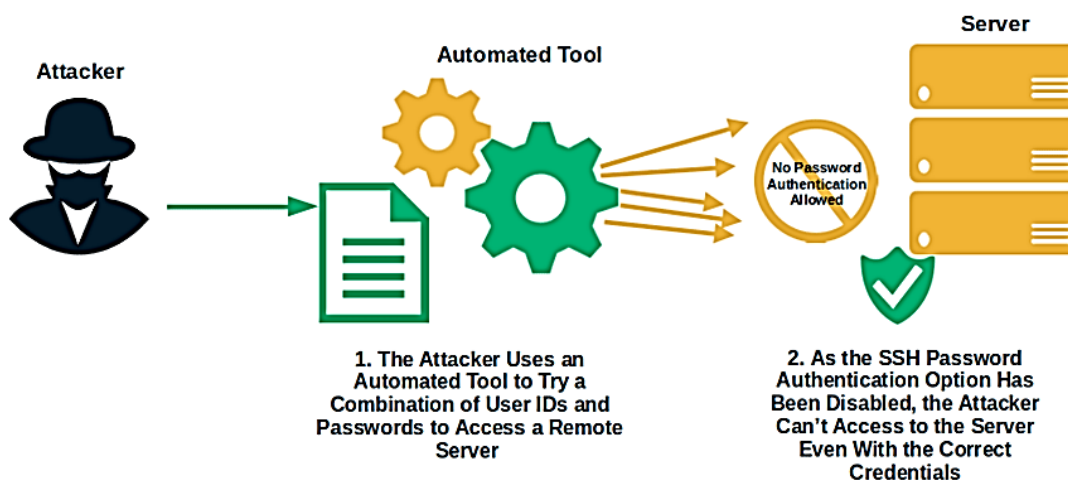


Master Password-less **SSH**: Complete Guide to Secure Remote Access from Windows to Linux VMs



How Passwordless SSH Can Protect Your Server From Brute Force Attacks



What is Passwordless SSH

Passwordless SSH (also known as key-based SSH authentication) is a secure method of logging into Linux systems without typing a password.

Instead of using traditional passwords, it relies on **cryptographic key pairs** (private key on the client and public key on the server) for authentication.

This approach eliminates the risk of brute-force password attacks and provides faster, automated, and more secure access to remote systems.

Why Passwordless SSH

- **Security Upgrade:** Protects against weak passwords and brute-force attacks since no password is transmitted over the network.
 - **Automation Friendly:** Ideal for scripts, DevOps pipelines, and system management tasks where manual password entry is not practical.
 - **Faster Access:** No need to type a password every time, saving time when frequently logging into servers.
 - **Centralized Trust Model:** Allows secure connection only if the client has the right private key.
 - **Compliance & Best Practice:** Recommended in enterprise environments to meet security standards.
-

Where we use Passwordless SSH

- **Server Administration:** Sysadmins/DevOps engineers connect to multiple servers quickly and securely.
- **Automated Deployments:** Tools like Ansible, Jenkins, or Kubernetes clusters rely on passwordless SSH for configuration and deployments.
- **Backup & File Transfers:** Secure, automated file transfer with rsync or scp without needing user intervention.
- **Cloud & Virtualization:** Widely used in AWS, GCP, and Azure environments for secure access to cloud servers.
- **Security Hardening:** Once keys are set, password login can be disabled, reducing attack vectors.

Ubuntu

Step 1: Verify SSH Client Installation on Windows.

```
PS C:\Users\UDDHAV> ssh -V
OpenSSH_for_Windows_9.5p1, LibreSSL 3.8.2
PS C:\Users\UDDHAV>
```

Purpose: Confirms that OpenSSH client is installed on your Windows machine and displays the version. This is essential before proceeding with SSH key generation and remote connections.

Why this command: Without SSH client installed, you cannot generate keys or connect to remote servers. This verification step ensures your Windows system is ready for SSH operations.

Step 2: Generate RSA Key Pair

```
PS C:\Users\UDDHAV\.ssh> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\UDDHAV\.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\UDDHAV\.ssh/id_rsa
Your public key has been saved in C:\Users\UDDHAV\.ssh/id_rsa.pub
The key fingerprint is:
SHA256:jc3XY2kfTNS7J/ztekQbInoK5yv0t0rOYvrB3H0oCTY uddhav@LAPTOP-462FPEKJ
The key's randomart image is:
+---[RSA 3072]---+
|                ..|
|                . .|
|                ..|
|      = . oo+.|
|      S = o.B++|
|    oo.o.o oo=+|
|    E.+=o.  ++|
|    . o+B+o.  +|
|    .++o=+.. .+.|
+-----[SHA256]-----+
PS C:\Users\UDDHAV\.ssh>
```

Why this command: Creates a public/private RSA key pair. The private key stays on your Windows machine, while the public key will be copied to VMs for authentication without passwords.

Step 3: Verify Generated SSH Keys

```
PS C:\Users\UDDHAV\.ssh> ls
```

```
Directory: C:\Users\UDDHAV\.ssh

Mode                LastWriteTime         Length Name
----                -
-a----           9/29/2025   9:46 AM             2610 id_rsa
-a----           9/29/2025   9:46 AM              577 id_rsa.pub
-a----           9/29/2025   9:26 AM            1476 known_hosts
-a----           9/29/2025   9:26 AM            1300 known_hosts.old
-a----           3/19/2025    9:15 PM            1462 new.ppk
-a----           4/10/2025    8:51 PM              98 windo.rdp
-a----           4/10/2025    8:57 PM              97 window.rdp
```

Why this command: (ls) Confirms that both the private key (id_rsa) and public key (id_rsa.pub) were successfully created. These are the essential files needed for passwordless authentication.

Step 4: Edit sshd configuration (enable root & pubkey auth)

```
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
```

What to change / check inside the file (ensure these lines exist or are uncommented):

- PermitRootLogin yes
- PubkeyAuthentication yes
- (optionally while testing) PasswordAuthentication yes
- later set PasswordAuthentication no after keys work

Why:

You must tell the SSH daemon to allow root login and accept public-key authentication.

PermitRootLogin yes allows root to log in (required if you want direct root SSH).

PubkeyAuthentication yes enables using authorized_keys. Keep PasswordAuthentication yes while setting up so you can still login with password if something breaks; disable it after verifying keys.

Step 5: Restart SSH to load new config

```
root@ubuntu:~# vi /etc/ssh/sshd_config
root@ubuntu:~# systemctl restart ssh
root@ubuntu:~# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2025-09-29 09:54:53 IST; 5s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Process: 2202 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 2203 (sshd)
     Tasks: 1 (limit: 5787)
    Memory: 1.0M
   CGroup: /system.slice/ssh.service
           └─2203 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Sep 29 09:54:53 ubuntu systemd[1]: Stopping OpenBSD Secure Shell server...
Sep 29 09:54:53 ubuntu systemd[1]: ssh.service: Succeeded.
Sep 29 09:54:53 ubuntu systemd[1]: Stopped OpenBSD Secure Shell server.
Sep 29 09:54:53 ubuntu systemd[1]: Starting OpenBSD Secure Shell server...
Sep 29 09:54:53 ubuntu sshd[2203]: Server listening on 0.0.0.0 port 22.
Sep 29 09:54:53 ubuntu sshd[2203]: Server listening on :: port 22.
Sep 29 09:54:53 ubuntu systemd[1]: Started OpenBSD Secure Shell server.
```

Why:

Changes in sshd_config only take effect after the SSH service is reloaded/restarted.

Confirm sshd is running and there are no startup errors (you'll see active (running) if OK).

```
root@ubuntu:~# exit
logout
uddhav@ubuntu:~$ exit
logout
Connection to 10.46.111.114 closed.
PS C:\Users\UDDHAV> █
```

Step 6: Generate SSH keypair on Windows (if not already)

On Windows PowerShell (run as your normal user, not root):

```
PS C:\Users\UDDHAV\.ssh> scp C:\Users\UDDHAV\.ssh\id_rsa.pub root@10.46.111.114
      1 file(s) copied.
PS C:\Users\UDDHAV\.ssh>
```

Why:

You need a public/private keypair. The public key goes to the server (authorized_keys), the private key stays on your Windows machine and is used to authenticate you.

Step 7: Copy the public key from Windows to the Ubuntu server

Using scp (as shown in your screenshots):

From PowerShell (example):

```
PS C:\Users\UDDHAV\.ssh> scp C:\Users\UDDHAV\.ssh\id_rsa.pub root@10.46.111.114:/root/.ssh/authorized_keys
root@10.46.111.114's password:
id_rsa.pub                                     100% 577   187.8KB/s   00:00
PS C:\Users\UDDHAV\.ssh> █
```

Why:

This copies your id_rsa.pub into /root/ on the Ubuntu server so you can append it to root's authorized_keys.

Step 8: Verify authorized_keys contains your key

```
root@ubuntu:~# cat /root/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC8Jah+ZF+fdq2KLMSiO4ohjm6zBiH8Yoj50Fn1EK0m043zaqDgvFLGcg7NdhEiTDqTc/NGxwBS
OqGSye5lB1xZHL9rg31vtB5kD1Tzw+kIm3mXUTdHS4EWJee26n8Q60gsBwWQs/vXX6dmPBwiU3quwBxXRY/70t+NfmiDMBSBviczxPhDKv4H9UwZ
6GugXPhEFO4kPYy3WZ3gGvDsCS8+FpvqJ3hLed/We4GIUhB6et9NlqvTgdnZ6lWjv29lMB8un4TRGCFQki1M8fMtkAGDXdJ9G8SR6aQyr91iKuw
FqgL646TfcNS7wIOKnPr5Th25dNO/aNLzwM0b/DR+0Tdf5LGAYxRncRkSAM47Ym9g25eDrKLRcXu08jqk2oBJEUxvm0A7/LIQyeCtYa2HmIoDTQy
9cG0C3P1+1kw+H+VX0fowu8yo+hU5Cj/IAAKHkVv3QrMm3463EwEEB3txwGKuFRdHkX7ckX4sHAWEx46CZmIVbHYQbRYCLcOMu6JrOE= uddhav@
LAPTOP-462FPFKJ
root@ubuntu:~# █
```

Set correct permissions (critical)

```
root@ubuntu:~# chmod 600 /root/.ssh/authorized_keys
root@ubuntu:~# chmod 700 /root/.ssh/
root@ubuntu:~#
```

Why:

OpenSSH is strict about permissions. The .ssh directory must be 700 (owner only), and authorized_keys must be 600 (read/write owner only). Wrong permissions will cause the SSH server to ignore authorized_keys.

Step 9: Harden SSHD config after successful key login

```
root@ubuntu:~# vi /etc/ssh/sshd_config
root@ubuntu:~# root@ubuntu:~#
```

Change to:

```
PasswordAuthentication no      # disable password auth entirely (if you're ready)
```

Then:

```
sudo systemctl restart ssh
```

```
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
```

Why:

Once key-based auth works, disable password authentication to prevent brute-force password attacks. prohibit-password (or without-password on older OpenSSH) allows root with publickey only.

Step 10: Take the SSH of VM from the Windows System.

```
PS C:\Users\UDDHAV> ssh root@10.46.111.114
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-139-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

* Introducing Expanded Security Maintenance for Applications.
  Receive updates to over 25,000 software packages with your
  Ubuntu Pro subscription. Free for personal use.

https://ubuntu.com/pro

Expanded Security Maintenance for Infrastructure is not enabled.
0 updates can be applied immediately.

Enable ESM Infra to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
root@ubuntu:~#
```

Why:

Look here we are able to login directly without entering the password.

This Password-Less Task has successfully done on the UBUNTU.

Now Let's check for the OpenSUSE.

OpenSUSE

Follow the all steps same as we did for the Ubuntu.

```
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
```

```
Connection to 10.46.111.196 closed.
PS C:\Users\UDDHAV> scp C:\Users\UDDHAV\.ssh\id_rsa.pub root@10.46.111.196:/root/.ssh/authorized_keys
(root@10.46.111.196) Password:
id_rsa.pub 100% 577 187.8KB/s 00:00
PS C:\Users\UDDHAV> _
```

```
opensuse:~ # systemctl restart sshd
opensuse:~ # systemctl status sshd
● sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: disabled)
   Active: active (running) since Mon 2025-09-29 10:15:23 IST; 4s ago
     Process: 10680 ExecStartPre=/usr/sbin/ssh-keygen -s /etc/ssh/ssh_host_rsa.pub (code=exited, status=0/SUCCESS)
     Process: 10682 ExecStartPre=/usr/sbin/ssh-keygen -s /etc/ssh/ssh_host_ecdsa.pub (code=exited, status=0/SUCCESS)
    Main PID: 10685 (sshd)
       Tasks: 1
          CPU: 28ms
      CGroup: /system.slice/ssh.service
              └─10685 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Sep 29 10:15:23 opensuse systemd[1]: Starting OpenSSH Daemon...
Sep 29 10:15:23 opensuse ssh-keygen[10680]: Checking for missing server keys in /etc/ssh
Sep 29 10:15:23 opensuse sshd[10685]: Server listening on 0.0.0.0 port 22.
Sep 29 10:15:23 opensuse sshd[10685]: Server listening on :: port 22.
Sep 29 10:15:23 opensuse systemd[1]: Started OpenSSH Daemon.
opensuse:~ # _
```

```
opensuse:~ # chmod 600 /root/.ssh/authorized_keys
opensuse:~ # chmod 700 /root/.ssh
opensuse:~ #
```

```
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
```

```
PS C:\Users\UDDHAV> ssh root@10.46.111.196
Have a lot of fun...
Last login: Mon Sep 29 10:16:45 2025 from 10.46.111.68
opensuse:~ #
```

Here also it didn't ask for the password...

