sshfs Mount over SSH

START HERE

`sshfs <user>@<host>:<dir> <mountpoint>`
Mount a remote dir

`sshfs <host>: <mountpoint>`
Mount remote home

`fusermount3 -u <mountpoint>`
Unmount (Linux)

`sshfs -p <port> <host>: <mountpoint>`
Non-default SSH port

`sshfs -o reconnect ...`
Survive drops

`sshfs -o ro ...`
Mount read-only

`sshfs -o allow_other ...`
Let other users in

`findmnt -t fuse.sshfs`
List sshfs mounts

Your programs ls, cp, editor → FUSE → **sshfs** local mountpoint → ssh + sftp → **sftp-server** on the remote host → disk → **Remote dir** /srv/data

CONCEPTS

sshfs speaks SFTP over a normal ssh login: if ssh <host> works, sshfs <host>:<dir> works. Run it as your own user, not root.

MOUNT AND UNMOUNT

`sshfs <user>@<host>:<dir> <mountpoint>`
Mount a remote directory

The mountpoint must exist and be owned by you. Without <user>@ your local name is used.

`sshfs <host>: <mountpoint>`
Mount the remote home directory

`fusermount3 -u <mountpoint>`
Unmount (Linux)

`umount <mountpoint>`
Unmount on macOS and BSD

`sshfs -p <port> <host>: <mountpoint>`
Use a non-default SSH port

`sshfs -o IdentityFile=<key> <host>: <mountpoint>`
Pick the key to log in with

`sshfs -F <ssh_config> <host>: <mountpoint>`
Use another ssh config file

`sshfs -o ro <host>: <mountpoint>`
Mount read-only

`sshfs -C <host>: <mountpoint>`
Compress traffic (slow links)

`findmnt -t fuse.sshfs`
List active sshfs mounts

`sshfs -f <host>: <mountpoint>`
Stay in the foreground

`sshfs -d <host>: <mountpoint>`
Debug output (implies -f)

`sshfs -o delay_connect <host>: <mountpoint>`
Connect on first access

`fusermount3 -uz <mountpoint>`
Lazy unmount a busy mount

`sshfs 'user@[<ipv6>]:<dir>' <mountpoint>`
Mount via a numeric IPv6 address

MOUNT AND UNMOUNT (cont.)

`sshfs -o directport=<port> <host>: <mountpoint>`
Skip ssh, talk to sftp-server directly

Unencrypted: only for localhost or trusted networks.

`sshfs -o ssh_command=<cmd> <host>: <mountpoint>`
Run another command instead of ssh

`sshfs -V`
Show sshfs and FUSE versions

RECONNECT AND HANGS

`sshfs -o reconnect,ServerAliveInterval=15 <host>: <mountpoint>`
Recover after the link drops

Files opened before the reconnect return errors and must be reopened.

`sshfs -o ServerAliveInterval=15,ServerAliveCountMax=3 <host>: <mountpoint>`
Drop a dead connection after ~45 s

`sshfs -o ConnectTimeout=10 <host>: <mountpoint>`
Fail fast when the host is down

`pkill -f 'sshfs.*<mountpoint>'`
Kill a hung sshfs, freeing stuck programs

Operations block forever by default; fusermount3 -uz does not unblock them, killing sshfs does.

> A frozen mount (ls hangs, cd stalls) usually means a dead connection with no keepalive: set ServerAliveInterval together with reconnect.

PERMISSIONS AND SYMLINKS

`sshfs -o allow_other <host>: <mountpoint>`
Let other local users use the mount

Non-root users need user_allow_other in /etc/fuse.conf.

`sshfs -o allow_other,default_permissions <host>: <mountpoint>`
Enforce file modes locally too

`sshfs -o idmap=user <host>: <mountpoint>`
Map remote user's IDs to yours

PERMISSIONS AND SYMLINKS (cont.)

`sshfs -o uid=<n>,gid=<n> <host>: <mountpoint>`
Show files as owned by this uid/gid

`sshfs -o umask=022 <host>: <mountpoint>`
Override file permissions shown

`sshfs -o idmap=file,uidfile=<f>,gidfile=<f> <host>: <mountpoint>`
Map IDs from name:id files

`sshfs -o idmap=file,nomap=ignore,uidfile=<f> <host>: <mountpoint>`
Skip names missing from the map

`sshfs -o follow_symlinks <host>: <mountpoint>`
Show remote symlinks as real files

`sshfs -o transform_symlinks <host>: <mountpoint>`
Rewrite absolute links as relative

`sshfs -o no_contain_symlinks <host>: <mountpoint>`
Allow absolute and .. link targets

Only with fully trusted servers.

SPEED AND CACHING

`sshfs -o dcache_timeout=<s> <host>: <mountpoint>`
Cache directory data longer (default 20 s)

`sshfs -o dir_cache=no <host>: <mountpoint>`
Always ask the server for fresh listings

`sshfs -o auto_cache <host>: <mountpoint>`
Reuse cached files until mtime changes

`sshfs -o kernel_cache <host>: <mountpoint>`
Keep file data in the kernel cache

`sshfs -o max_conns=<n> <host>: <mountpoint>`
Use several ssh connections

`sshfs -o workaround=rename <host>: <mountpoint>`
Allow mv over an existing file

`sshfs -o workaround=renamexdev <host>: <mountpoint>`
Let mv fall back to copy across mounts

SPEED AND CACHING (cont.)

`sshfs -o sshfs_sync <host>: <mountpoint>`
Synchronous writes: safer, slower

`sshfs -o sync_readdir <host>: <mountpoint>`
Synchronous directory reads

> Chatty workloads (git status, builds, find) are slow over sshfs: raise dcache_timeout, or work on the remote host directly.

MOUNTING AT BOOT

`sudo mount <mountpoint>`
Mount an fstab entry by mount point

`mkdir -p <mountpoint>`
Create the mountpoint first

fstab line

```
<user>@<host>:<dir>
<mountpoint> fuse.sshfs \
  _netdev,reconnect,
ServerAliveInterval=15,
IdentityFile=<key> 0 0
# one line in the real file
```

> fstab mounts run as root, so give an absolute IdentityFile path that root can read, and accept the host key once as root first.

> Add noauto,x-systemd.automount to mount on first access instead of at boot.

RECIPES

Mount a server for the day

```
mkdir -p ~/mnt/box
sshfs -o reconnect,
ServerAliveInterval=15
<user>@<host>:/srv/data ~/mnt/
box
ls ~/mnt/box
fusermount3 -u ~/mnt/box
```

Read-only mount on a custom port

```
mkdir -p ~/mnt/logs
sshfs -p 2222 -o ro,
IdentityFile=~/.ssh/id_ed25519
<host>:/var/log ~/mnt/logs
```

Share a mount with other users

```
echo user_allow_other | sudo
tee -a /etc/fuse.conf
sshfs -o allow_other,
default_permissions,idmap=user
<host>:/srv/data /mnt/data
```

Recover a hung mount

```
pkill -f 'sshfs.*mnt/box'
fusermount3 -uz ~/mnt/box
sshfs -o reconnect,
ServerAliveInterval=15 <host>:
~/mnt/box
```