Table of Contents

$ rsync

Intermediate

Fast and versatile file synchronization tool

rsync is a fast, versatile file copying tool that synchronizes files and directories between local and remote locations. It is the gold standard for backups, deployments, and file synchronization because it only transfers changed portions of files.

rsync uses a delta-transfer algorithm to send o...

Options & Flags

`-a`	Archive mode (preserves permissions, timestamps, symlinks...
`-v`	Verbose output
`-z`	Compress data during transfer
`--delete`	Delete files in destination not in source
`-n`	Dry run - show what would be done without doing it
`--exclude`	Exclude files matching pattern
`-P`	Show progress and enable partial (resume)
`-e`	Specify remote shell (e.g., SSH with custom port)
`--backup`	Make backups of replaced files

Practical Examples

Example: Sync directories locally

```
$ rsync -av /var/www/html/ /backup/www/
```

Copies the web root to backup, preserving all attributes. Note the trailing slash on source.

Example: Sync to remote server

```
$ rsync -avz /var/www/ user@server:/var/www/
```

Synchronizes the web directory to a remote server with compression.

Example: Backup with deletion

```
$ rsync -av --delete /data/ /backup/data/
```

Creates an exact mirror - files deleted from source are also deleted from backup.

Example: Dry run before sync

```
$ rsync -avn --delete /src/ /dest/
```

Shows what would be transferred or deleted without actually doing anything.

Example: Exclude patterns

```
$ rsync -av --exclude='*.log' --exclude='node_modules' /project/ /backup/
```

Syncs project excluding log files and node_modules.

Tips & Best Practices

Warning: Trailing slash matters: rsync /src/ /dest/ copies contents of src into dest. rsync /src /dest/ copies the src directory itself into dest. The trailing slash on source changes behavior significantly.

Pro Tip: Always dry-run first: Use rsync -avn (or --dry-run) before any sync with --delete to preview what will be removed. This prevents accidental data loss.

Note: Incremental backups: Use --link-dest for space-efficient incremental backups: rsync -a --link-dest=/backup/yesterday /data/ /backup/today/

$ scp

Beginner

Securely copy files between hosts over SSH

scp (secure copy) transfers files between local and remote hosts over SSH. It uses the same authentication and encryption as SSH, making it a secure alternative to FTP for file transfers.

scp supports copying files in both directions: from local to remote, from remote to local, and between two r...

Options & Flags

`-r`	Copy directories recursively
`-P`	Specify SSH port
`-i`	Use specific identity key
`-C`	Enable compression during transfer
`-p`	Preserve timestamps and permissions
`-q`	Quiet mode (no progress bar)
`-l`	Limit bandwidth in Kbit/s

Practical Examples

Example: Copy file to remote server

```
$ scp report.pdf user@server:/home/user/Documents/
```

Uploads a file to the remote server.

Example: Copy file from remote server

```
$ scp user@server:/var/log/app.log ./app.log
```

Downloads a file from the remote server to local machine.

Example: Copy directory recursively

```
$ scp -r ./deploy user@server:/var/www/html/
```

Copies an entire directory tree to the remote server.

Example: Copy with custom port

```
$ scp -P 2222 config.yml admin@server:/etc/app/
```

Copies a file using a non-standard SSH port.

Example: Copy between two remote hosts

```
$ scp user@server1:/data/db.sql user@server2:/backup/
```

Copies a file directly between two remote servers.

Tips & Best Practices

Pro Tip: Use rsync instead for large transfers: rsync -avz is superior to scp for large directories - it only transfers changed files and can resume interrupted transfers.

Note: Use SSH config for shortcuts: Define hosts in ~/.ssh/config, then use: scp file.txt myserver:/path/ instead of typing full user@hostname.

Warning: SCP protocol deprecated: The SCP protocol has known limitations. OpenSSH 9+ uses SFTP internally for scp. For new scripts, consider using rsync or sftp directly.

$ sftp

Intermediate

Secure File Transfer Protocol (interactive)

sftp (SSH File Transfer Protocol) provides interactive file transfer over an encrypted SSH connection. It offers an FTP-like interface with the security of SSH, supporting file uploads, downloads, directory navigation, and remote file management.

Unlike FTP which sends data in clear text, sftp e...

Options & Flags

`-P`	Connect to specific port
`-i`	Use specific identity key
`-b`	Batch mode - read commands from file
`-r`	Recursive operations (used with get/put)
`-C`	Enable compression
`-l`	Limit bandwidth in Kbit/s

Practical Examples

Example: Connect to server

```
$ sftp user@server.example.com
sftp>
```

Opens interactive SFTP session. Type help for available commands.

Example: Upload a file

```
$ sftp> put report.pdf /var/www/uploads/
```

Uploads a local file to the remote server.

Example: Download a file

```
$ sftp> get /var/log/app.log ./local-copy.log
```

Downloads a remote file to the local machine.

Example: Upload directory recursively

```
$ sftp> put -r ./project /var/www/html/
```

Uploads an entire directory tree.

Example: Navigate and list

```
$ sftp> cd /var/www\nsftp> ls -la
```

Navigate and list files on the remote server.

Tips & Best Practices

Pro Tip: Common SFTP commands: put (upload), get (download), ls (list), cd (change dir), lcd (change local dir), mkdir, rm, rename, bye/exit (quit).

Note: sftp vs scp vs rsync: sftp is interactive (browse and select files). scp is for quick non-interactive copies. rsync is for incremental synchronization. Choose based on your workflow.

Warning: Not the same as FTPS: sftp (SSH FTP) and FTPS (FTP over SSL) are completely different protocols. sftp uses SSH port 22; FTPS uses FTP ports 21/990.

$ ssh

Beginner

Securely connect to remote servers

SSH (Secure Shell) provides encrypted remote login and command execution over insecure networks. It is the primary method for securely accessing remote Linux servers, replacing older unencrypted protocols like telnet and rsh.

ssh supports password authentication, public key authentication, X11 f...

Options & Flags

`-p`	Connect to specific port
`-i`	Use specific identity (private key) file
`-L`	Local port forwarding (tunnel)
`-R`	Remote port forwarding
`-D`	Dynamic port forwarding (SOCKS proxy)
`-N`	Do not execute remote command (tunneling only)
`-v`	Verbose mode for debugging
`-A`	Enable agent forwarding
`-J`	Jump host (proxy through another server)
`-t`	Force pseudo-terminal allocation

Practical Examples

Example: Connect to a server

```
$ ssh user@192.168.1.100
```

Opens an interactive shell on the remote server.

Example: Run remote command

```
$ ssh user@server "df -h && free -m"
```

Executes commands on the remote server and returns output locally.

Example: SSH with key file

```
$ ssh -i ~/.ssh/mykey.pem ubuntu@ec2-instance.aws.com
```

Connects using a specific SSH private key file.

Example: Port forwarding (tunnel)

```
$ ssh -L 3306:localhost:3306 user@db-server -N
```

Forwards local port 3306 to the database server - access remote MySQL locally.

Example: Jump through bastion host

```
$ ssh -J admin@bastion user@internal-server
```

Connects to an internal server through a bastion/jump host.

Tips & Best Practices

Pro Tip: Use SSH config file: Create ~/.ssh/config to define shortcuts: Host myserver / Hostname 192.168.1.100 / User admin / IdentityFile ~/.ssh/mykey. Then just type: ssh myserver

Warning: Never use password auth in production: Always use key-based authentication. Disable password auth in /etc/ssh/sshd_config: PasswordAuthentication no

Note: SSH agent for key management: Use ssh-agent and ssh-add to cache your passphrase: eval $(ssh-agent) && ssh-add ~/.ssh/id_rsa. Avoids entering passphrase repeatedly.

$ ssh-copy-id

Intermediate

Install your SSH public key on a remote server

ssh-copy-id installs your SSH public key on a remote server, enabling passwordless SSH authentication. It appends your public key to the remote server's ~/.ssh/authorized_keys file.

ssh-copy-id handles all the details: creating the .ssh directory if needed, setting correct permissions (700 for ....

Options & Flags

`-i`	Specify identity file to copy
`-p`	Specify SSH port
`-o`	Pass SSH options

Practical Examples

Example: Copy default key

```
$ ssh-copy-id user@192.168.1.100
Number of key(s) added: 1
Now try: ssh 'user@192.168.1.100'
```

Copies the default public key to the server. Prompts for password one last time.

Example: Copy specific key

```
$ ssh-copy-id -i ~/.ssh/deploy_key.pub deploy@server.com
```

Copies a specific key to the server.

Example: Custom SSH port

```
$ ssh-copy-id -p 2222 admin@server.example.com
```

Copies key to a server running SSH on port 2222.

Example: Complete setup

```
$ ssh-keygen -t ed25519 && ssh-copy-id user@server && ssh user@server
```

Full workflow: generate key, copy to server, connect passwordless.

Example: Manual alternative

```
$ cat ~/.ssh/id_ed25519.pub | ssh user@server "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys && chmod 600 ~/.ssh/authorized_keys
```

Manual method if ssh-copy-id is not available.

Tips & Best Practices

Pro Tip: One-time password prompt: ssh-copy-id asks for the password ONE last time to install the key. After that, SSH is passwordless.

Note: Sets correct permissions: ssh-copy-id automatically sets ~/.ssh to 700 and authorized_keys to 600 on the remote server.

Warning: Disable password auth after: After setting up key auth, disable password authentication in /etc/ssh/sshd_config: PasswordAuthentication no.

$ ssh-keygen

Intermediate

Generate SSH key pairs for authentication

ssh-keygen generates, manages, and converts SSH authentication keys. It creates public-private key pairs used for passwordless SSH login, Git authentication, and secure communication.

ssh-keygen supports multiple key types: RSA, Ed25519 (recommended), ECDSA, and DSA (deprecated). Ed25519 keys ar...

Options & Flags

`-t`	Key type (ed25519, rsa, ecdsa)
`-C`	Comment (usually email)
`-f`	Output filename
`-b`	Key bits (for RSA)
`-p`	Change passphrase
`-R`	Remove host from known_hosts
`-l`	Show key fingerprint

Practical Examples

Example: Generate Ed25519 key

```
$ ssh-keygen -t ed25519 -C "user@example.com"
Generating public/private ed25519 key pair.\nEnter file: ~/.ssh/id_ed25519
```

Creates an Ed25519 key pair - the recommended type.

Example: Generate RSA key

```
$ ssh-keygen -t rsa -b 4096 -C "user@example.com"
```

Creates a 4096-bit RSA key pair.

Example: Generate without passphrase

```
$ ssh-keygen -t ed25519 -f ~/.ssh/deploy_key -N ""
```

Creates a key without passphrase - use only for automation.

Example: Show fingerprint

```
$ ssh-keygen -l -f ~/.ssh/id_ed25519.pub
256 SHA256:abc123... user@example.com (ED25519)
```

Displays the key fingerprint for verification.

Example: Change passphrase

```
$ ssh-keygen -p -f ~/.ssh/id_ed25519
```

Changes the passphrase on an existing key.

Tips & Best Practices

Pro Tip: Use Ed25519: Ed25519 is recommended: faster, shorter keys, and more secure than RSA. ssh-keygen -t ed25519.

Warning: Protect private keys: chmod 600 ~/.ssh/id_ed25519. Never share private keys. Use a passphrase unless automated (deploy keys).

Note: Multiple keys: Use -f to create named keys for different purposes: ssh-keygen -f ~/.ssh/github_key. Configure in ~/.ssh/config.