gzip, xz and zstd Compress files

START HERE

`gzip` *<file>*
Compress, replacing the file

`zstd -d` *<file.zst>*
Decompress zstd

`gzip -d` *<file.gz>*
Decompress

`zcat` *<file.gz>*
Read without unpacking

`xz -k` *<file>*
Smaller, slower, keep original

`gzip -t` *<file.gz>*
Test integrity

`zstd` *<file>*
Fast and good, keeps original

`tar -caf` *<out.tar.zst>* *<dir>*
Compress a whole directory

gzip fast, universal → smaller

zstd fast, modern → smaller, slower

xz smallest, slow

CONCEPTS

gzip and xz delete the input file by default; zstd keeps it. Add -k to gzip/xz, or --rm to zstd to delete.

These tools compress single streams, not folders. Use tar to bundle a directory first (tar -caf picks the compressor from the suffix).

Rule of thumb: gzip for compatibility, zstd for speed and everyday use, xz for archives that are written once and downloaded often.

Higher levels cost far more time than they save space. xz -7 to -9 also need much more memory to decompress.

Already-compressed data (jpg, mp4, zip, .gz) barely shrinks; compressing it again just wastes time.

GZIP

`gzip` *<file>*
Compress to <file>.gz, remove original

`gzip -k` *<file>*
Compress, keep the original

`gzip -d` *<file.gz>*
Decompress (same as gunzip)

`gzip -c` *<file>* `>` *<out.gz>*
Write to stdout, keep original

`gzip -9` *<file>*
Best compression (slowest)
Levels are -1 (fast) to -9 (best); the default is -6.

`gzip -1` *<file>*
Fastest compression

`gzip -r` *<dir>*
Compress every file below a dir
Each file becomes its own .gz; it does not make an archive.

`gzip -n` *<file>*
Do not store name and timestamp
Makes output reproducible for identical input.

XZ

`xz` *<file>*
Compress to <file>.xz, remove original

`xz -k` *<file>*
Compress, keep the original

`xz -d` *<file.xz>*
Decompress (same as unxz)

XZ (cont.)

`xz -c` *<file>* `>` *<out.xz>*
Write to stdout, keep original

`xz -9` *<file>*
Strong compression, high memory
Presets -0 to -9; default is -6.

`xz -9e` *<file>*
Extreme: a bit smaller, much slower

`xz -T0` *<file>*
Use all CPU cores
Threaded output is split into blocks, so it is slightly larger. In xz 5.8 the default is already -T0.

`xz -T`*<n>* *<file>*
Use at most n threads

ZSTD

`zstd` *<file>*
Compress to <file>.zst, keep original

`zstd -d` *<file.zst>*
Decompress (same as unzstd)

`zstd -o` *<out>* *<file>*
Choose the output name

`zstd -c` *<file>* `>` *<out.zst>*
Write to stdout

`zstd --rm` *<file>*
Delete input after success

`zstd -19` *<file>*
High compression, slower
Levels 1-19, default 3. --ultra unlocks 20-22 and needs lots of memory.

`zstd --fast=`*<n>* *<file>*
Faster than level 1, larger output

`zstd -T0` *<file>*
Use all CPU cores
Default is a single thread.

`zstd --long=27` *<file>*
Long-distance matching, 128 MB window
Helps with big files with far-apart repeats. The decompressor needs --long=27 too (or --memory) for windows over 128 MB.

`zstd --ultra -22` *<file>*
Maximum compression

`zstd --adapt` *<file>*
Tune level to I/O speed
Good when piping to a slow network.

`zstd --train` *<files>* `-o` *<dict>*
Train a dictionary
Mainly useful for many small, similar files.

`zstd -b`*<n>* *<file>*
Benchmark level n on a file

`zstd -f` *<file>*
Overwrite existing output

READ AND INSPECT

`zcat` *<file.gz>*
Print decompressed content

`xzcat` *<file.xz>*
Print decompressed content

`zstdcat` *<file.zst>*
Print decompressed content

`zcat` *<file.gz>* `| less`
Page through a compressed file

`zgrep` *<pat>* *<file.gz>*
Search without unpacking
xzgrep and zstdgrep work the same way.

`zless` *<file.gz>*
Browse a compressed file

`gzip -l` *<file.gz>*
Show sizes and ratio
The uncompressed size is wrong for files over 4 GiB.

`xz -l` *<file.xz>*
Show sizes, check and blocks

`zstd -l` *<file.zst>*
Show frames, sizes, ratio

TEST AND CHECK

`gzip -t` *<file.gz>*
Test integrity, silent if ok

`xz -t` *<file.xz>*
Test integrity

`zstd -t` *<file.zst>*
Test integrity

`gzip -tv` *<file.gz>*
Test and print result

PIPES AND ARCHIVES

<cmd> **`|`** **`gzip >`** *<out.gz>*
Compress a stream

`gzip -dc` *<file.gz>* `|` *<cmd>*
Decompress into a pipe

<cmd> **`| zstd -T0 | ssh`** *<host>* **`'cat >`** *<out.zst>`**
Compress before sending

`tar -caf` *<out.tar.zst>* *<dir>*
Archive, compressor from suffix

`tar -I 'zstd -T0 -19' -cf` *<out.tar.zst>* *<dir>*
Archive with custom options

`tar -xf` *<out.tar.zst>*
Extract any tar compression

`tar -I pigz -cf` *<out.tar.gz>* *<dir>*
Parallel gzip if pigz is installed

ZIP AND UNZIP

`zip -r` *<out.zip>* *<dir>*
Zip a directory

`zip` *<out.zip>* *<file>`...`*
Zip some files

`unzip` *<file.zip>*
Extract here

`unzip` *<file.zip>* `-d` *<dir>*
Extract into a directory

`unzip -l` *<file.zip>*
List contents

`unzip -t` *<file.zip>*
Test integrity

`unzip` *<file.zip>* *<member>*
Extract one member

`zip -e` *<out.zip>* *<file>*
Password-protect
The classic zip encryption is weak; use age or gpg for secrets.

`zip -r` *<out.zip>* *<dir>* `-x '*.git*'`
Exclude patterns

WHICH FORMAT

Compared

Tool	Speed	Size	Use for
`gzip`	fast	medium	compatibility, logs
`zstd`	very fast	good	everyday, backups
`xz`	slow	smallest	release tarballs
`zip`	fast	medium	sharing with Windows

Decompression is much faster than compression for all three, and zstd decompresses fastest regardless of level.

RECIPES

Max-compress a big file using all cores

```
zstd -19 -T0 --long=27 big.img
# decompress: zstd -d --long=27
big.img.zst
```

Compress a database dump on the fly

```
pg_dump mydb | zstd -T0 >
mydb.sql.zst
zstdcat mydb.sql.zst | psql
mydb
```

Compress every log older than a day

```
find /var/log/app -name '*.log'
-mtime +1 \
  -exec gzip {} +
```

Compare formats on your data

```
for c in 'gzip -9' 'zstd -19'
'xz -9'; do
  printf '%s: ' "$c"; $c -c
data | wc -c
done
```