zstd Fast compression

START HERE

zstd *<file>*
Compress to <file>.zst

tar --zstd -cf *<a>*.**tar.zst** *<dir>*
Archive a directory

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

zstd --rm *<file>*
Delete original after

zstd -19 *<file>*
Smaller, slower

zstd -T0 *<file>*
Use all cores

zstd -l *<file>*.**zst**
Show sizes and ratio

zstd -dc *<file>*.**zst**
Print to stdout

CONCEPTS

zstd keeps the input file by default and adds .zst. Use --rm to delete the original once it succeeds.

--fast=N fastest, largest
↓ -1
-3 default balance
↓ -19
--ultra -22 smallest, slow, lots of RAM
Level versus speed

COMPRESS

zstd *<file>*
Compress, keep original (<file>.zst)

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

zstd -<*n*> *<file>*
Level 1-19 (default 3)

zstd -c *<file>* **>** *<out>*
Write to stdout, keep input

zstd --rm *<file>*
Remove input after success

zstd -f *<file>*
Overwrite existing output

zstd -r *<dir>*
Compress each file in a directory
Makes one .zst per file, not one archive. Use tar for a single archive.

zstd *<a>* *<b>* *<c>*
Compress several files at once

zstd --fast=<*n*> *<file>*
Faster than level 1, lower ratio

COMPRESS (cont.)

zstd --ultra -22 *<file>*
Levels 20-22, heavy memory use
Decompression then needs lots of memory too, so check the target machine.

zstd --adapt *<file>*
Auto-tune level to I/O speed

zstd --exclude-compressed -r *<dir>*
Skip already compressed files

zstd --no-check *<file>*
Skip the XXH64 checksum

zstd --format=gzip *<file>*
Write gzip, xz, lzma or lz4 instead

DECOMPRESS

zstd -d *<file>*.**zst**
Decompress, keep the .zst

unzstd *<file>*.**zst**
Same as zstd -d

zstd -dc *<file>*.**zst**
Decompress to stdout

zstdcat *<file>*.**zst**
Print a compressed file

zstdless *<file>*.**zst**
Page a compressed file

zstdgrep *<pattern>* *<file>*.**zst**
Search inside a compressed file

zstd -d *<file>*.**zst -o** *<out>*
Choose the output name

DECOMPRESS (cont.)

zstd -df *<file>*.**zst**
Overwrite existing output

zstd -d -M<*mb*> *<file>*.**zst**
Cap decompression memory

zstd --test *<file>*.**zst**
Verify integrity only

zstd -d --long=<*n*> *<file>*.**zst**
Needed if made with a large --long window
Windows above 27 bits must be allowed explicitly on decompression.

INSPECT

zstd -l *<file>*.**zst**
Frames, sizes, ratio, checksum

zstd -lv *<file>*.**zst**
Verbose: window size, dictionary ID

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

zstd -b1 -e19 *<file>*
Benchmark a range of levels

zstd -v *<file>*
Verbose output

SPEED AND MEMORY

zstd -T0 *<file>*
Use all physical cores

zstd -T<*n*> *<file>*
Use n threads

SPEED AND MEMORY (cont.)

zstd --auto-threads=logical -T0 *<file>*
Count logical cores for -T0

zstd --long *<file>*
Long-range matching for big, repetitive data

zstd --long=<*n*> *<file>*
Window of 2^n bytes (default 27)

zstd --rsyncable *<file>*
Output that rsync can delta well

zstd --single-thread *<file>*
One thread for I/O and compression

zstd -B<*size*> *<file>*
Set job size per thread

zstd --size-hint=<*bytes*>
Tune for piped input of known size

PIPES AND TAR

tar --zstd -cf *<a>*.**tar.zst** *<dir>*
Create a tar.zst

tar --zstd -xf *<a>*.**tar.zst**
Extract a tar.zst

tar -tf *<a>*.**tar.zst**
List contents
Modern GNU tar detects the zstd format on read, so --zstd is only needed when creating.

PIPES AND TAR (cont.)

tar -I 'zstd -T0 -19' -cf *<a>*.**tar.zst** *<dir>*
Pick level and threads for tar

tar -cf - <dir> | zstd -T0 > <a>.tar.zst
Pipe tar into zstd

zstd -dc *<a>*.**tar.zst | tar -xf -**
Pipe zstd into tar

zstdmt *<file>*
Alias for multithreaded zstd

zstd -c *<file>* **| ssh** *<host>* **'zstd -d >** *<out>***'**
Compress over ssh

DICTIONARIES AND PATCHES

zstd --train *<files>* **-o** *<dict>*
Build a dictionary from samples
Needs many small, similar sample files to work well.

zstd --train --maxdict=<*bytes*> *<files>*
Limit the dictionary size

zstd -D *<dict>* *<file>*
Compress with a dictionary

zstd -D *<dict>* **-d** *<file>*.**zst**
Decompress with the same dictionary

zstd --patch-from=<*old*> *<new>* **-o** *<patch>*
Make a binary diff against old

DICTIONARIES AND PATCHES (cont.)

zstd -d --patch-from=<*old*> *<patch>* **-o** *<new>*
Apply that patch to old

zstd --filelist *<list>*
Read file names from a list

zstd --output-dir-flat *<dir>* *<files>*
Put outputs in one directory

zstd --output-dir-mirror *<dir>* **-r** *<src>*
Mirror the source tree structure

RECIPES

Fast backup archive

```
tar -I 'zstd -T0 -10' -cf
backup.tar.zst <dir>
zstd --test backup.tar.zst
zstd -l backup.tar.zst
```

Maximum compression for a release

```
zstd --ultra -22 -T0 --long=27
<file> -o <file>.zst
# decompress with the same
window
zstd -d --long=27 <file>.zst
```

Dump a database through zstd

```
pg_dump <db> | zstd -T0 >
<db>.sql.zst
zstd -dc <db>.sql.zst | psql
<db>
```

Dictionary for many small files

```
zstd --train logs/*.json -o
logs.dict
zstd -D logs.dict logs/*.json
zstd -D logs.dict -d logs/*.zst
```