zstd Fast compression

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

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

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

zstd --ultra -22 *<file>*
Levels 20-22, heavy memory use

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

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

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

DECOMPRESS (cont.)

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

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

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

SPEED AND MEMORY

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

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

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

PIPES AND TAR

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

PIPES AND TAR (cont.)

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

tar -tf *<a>***.tar.zst**
List contents

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

DICTIONARIES AND PATCHES

zstd --train *<files>* **-o** *<dict>*
Build a dictionary from samples

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

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

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
```