Memcached In-memory cache

START HERE

`memcached -m 256 -l 127.0.0.1`
Run with 256 MB, local only

`nc <host> 11211`
Open a text session

`set <key> <flags> <ttl> <bytes>`
Store a value

`get <key>`
Read it back

`delete <key>`
Remove it

`incr <key> <n>`
Atomic counter

`stats`
Hit rate, memory, evictions

`flush_all`
Invalidate everything

Client hashes key to a server → `get <key>` → **memcached** slabs + LRU → `miss` → **Database** source of truth → `set <key>` → **memcached** next read hits

CONCEPTS

Memcached is a plain in-memory key-value cache: no persistence, no auth by default, and old items are evicted when memory is full. Never expose it to the internet.

Servers know nothing about each other: the client library picks the server per key, so adding nodes adds capacity.

A ttl of 0 means no expiry, but items can still be evicted; a ttl above 30 days is read as a Unix timestamp.

RUN THE DAEMON

`memcached -m <MB>`
Set item memory (default 64)
Not a hard cap; real usage runs a bit higher. Check maxbytes in stats settings.

`memcached -p <port>`
TCP port (default 11211)

`memcached -l <addr>`
Listen on one interface
Bind to 127.0.0.1 or a private address, never a public one.

`memcached -d -u <user>`
Daemonize, drop to a user
-u only applies when started as root.

`memcached -c <n>`
Max connections (default 1024)

`memcached -t <n>`
Worker threads (default 4)

`memcached -v / -vv`
Log errors / also commands

`memcached -s <path>`
Listen on a UNIX socket
Disables TCP. Combine with -a <mask> for permissions.

`memcached -U <port>`
Enable UDP (off by default)

`memcached -M`
Error on full instead of evicting

`memcached -I <size>`
Max item size (default 1m)

`memcached -P <file>`
Write a pid file (with -d)

`memcached -F`
Disable flush_all

`memcached -A`
Enable the shutdown command

RUN THE DAEMON (cont.)

`memcached -B ascii`
Force the text protocol

`memcached -o <opt>,<opt>`
Extended options

`memcached -V`
Print version

TALK TO IT

`nc <host> <port>`
Open a raw text session
Type commands ending in Enter; nc usually sends \n, which memcached accepts.

`printf '<cmd>\r\n' | nc -q1 <host> <port>`
Send one command
-q1 is the traditional netcat option; OpenBSD nc uses -N.

`telnet <host> <port>`
Same, with telnet

`version`
Show server version

`verbosity <n>`
Change log level at runtime

`quit`
Close the session

STORE AND FETCH

`set <key> <flags> <ttl> <bytes>`
Store, overwrite if present
The value goes on the next line and must be exactly <bytes> long. Reply: STORED.

`get <key> [<key> ...]`
Fetch one or more keys
Misses are simply absent from the reply; END closes it.

`delete <key>`
Remove a key

`add <key> <flags> <ttl> <bytes>`
Store only if absent
Replies NOT_STORED if the key exists; useful as a simple lock.

`replace <key> <flags> <ttl> <bytes>`
Store only if present

`append <key> 0 0 <bytes>`
Add bytes after the value

`prepend <key> 0 0 <bytes>`
Add bytes before the value

`gets <key>`
Fetch with a CAS id

STORE AND FETCH (cont.)

`cas <key> <flags> <ttl> <bytes> <cas>`
Store only if unchanged
EXISTS means someone changed it since your gets.

`touch <key> <ttl>`
Update expiry only

`gat <ttl> <key>`
Get and set new expiry

`gats <ttl> <key>`
Get, touch, return CAS

COUNTERS

`incr <key> <n>`
Add n to a number
The value must be a decimal string; replies with the new number.

`decr <key> <n>`
Subtract n, floor at 0

`set <key> 0 0 1`
Seed a counter with 0
Then send 0 on the next line. incr on a missing key gives NOT_FOUND.

META COMMANDS

`mg <key> v`
Meta get: return value

`mg <key> v t f c k`
Value plus ttl, flags, CAS, key
Each flag adds a field to the reply; t is -1 for no expiry.

`ms <key> <bytes> T<ttl>`
Meta set with a ttl

`md <key>`
Meta delete

`mn`
No-op, marks end of a pipeline

`mg <key> s v q`
Quiet get: no reply on miss

`ma <key>`
Meta arithmetic (incr)

STATS AND INSPECTION

`stats`
General counters

`stats settings`
Effective config of the daemon

`stats items`
Per-slab item counts and age

`stats slabs`
Per-slab memory use

STATS AND INSPECTION (cont.)

`stats sizes`
Item size histogram
Needs -o track_sizes on current versions.

`stats conns`
Open connections

`stats reset`
Zero resettable counters

`lru_crawler metadump all`
List keys with metadata
No plain keys command exists; this is the supported way. Disabled by -X.

`watch`
Stream live log of requests

`stats cachedump <slab> <n>`
Dump keys of a slab
Legacy; limited output.

`flush_all [<delay>]`
Invalidate all items
Items are only marked stale, memory is not freed immediately.

`shutdown`
Stop the server

KEY STATS TO READ

Fields in stats

Field	Look for
`get_hits / get_misses`	hits / (hits+misses)
`evictions`	rising: too little RAM
`curr_items`	items now
`bytes / limit_maxbytes`	used vs. limit
`curr_connections`	clients now
`listen_disabled_num`	non-zero: -c hit

RECIPES

Round trip from the shell

```
printf 'set k 0 60 5\r\nhello\r\nget k\r\nquit\r\n' | nc -q1 127.0.0.1 11211
```

Hit ratio

```
printf 'stats\r\nquit\r\n' | nc -q1 127.0.0.1 11211 |
  awk '/get_hits/{h=$3} /get_misses/{m=$3} END{print h/(h+m)}'
```

Dump all keys

```
printf 'lru_crawler metadump all\r\nquit\r\n' | nc -q1 127.0.0.1 11211 | cut -d' ' -f1
```

Throwaway local cache

```
memcached -m 64 -l 127.0.0.1 -p 11311 -vv
```