Memcached In-memory cache

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.

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

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

RUN THE DAEMON

`memcached -m` *`<MB>`*
Set item memory (default 64)

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

`memcached -l` *`<addr>`*
Listen on one interface

`memcached -d -u` *`<user>`*
Daemonize, drop to a user

`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

TALK TO IT

`nc` *`<host> <port>`*
Open a raw text session

`printf '<cmd>\r\n' | nc -q1` *`<host> <port>`*
Send one command

`version`
Show server version

STORE AND FETCH

`set` *`<key> <flags> <ttl> <bytes>`*
Store, overwrite if present

`get` *`<key>`* **`[`***`<key>`***` ...]`**
Fetch one or more keys

`delete` *`<key>`*
Remove a key

`add` *`<key> <flags> <ttl> <bytes>`*
Store only if absent

`gets` *`<key>`*
Fetch with a CAS id

`cas` *`<key> <flags> <ttl> <bytes> <cas>`*
Store only if unchanged

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

COUNTERS

`incr` *`<key> <n>`*
Add n to a number

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

META COMMANDS

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

STATS AND INSPECTION

`stats`
General counters

`stats settings`
Effective config of the daemon

`flush_all [`*`<delay>`***`]`**
Invalidate all items

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