

## **Memcached Cheatsheet**

A quick reference for Memcached, covering basic commands, data management, and common use cases. Ideal for developers needing a fast, distributed memory object caching system.



### **Core Concepts & Commands**

#### **Basic Operations**

Stores data under the specified key. Flags are set <key> <flags> <exptime> arbitrary 16-bit integer, exptime is expiration time in seconds (O means never expire), bytes is data length. <bytes>\r\n<data>  $r\n$ Example: set mykey 0 3600 5\r\nvalue\r\n Retrieves data associated with the specified key. get <key>\r\n get mykey\r\n

> Stores data under the specified key, but only if the server doesn't already hold data for this key.

> Stores data under the specified key, but only if the

replace existingkey 0 3600 5\r\nvalue\r\n

Decrements the value of the key by value. The value

add newkey 0 3600 5\r\nvalue\r\n

server does already hold data for this key.

Example:

Example:

add <key> <flags> <exptime> <bytes>\r\n<data>  $r\n$ 

replace <key>

<flags> <exptime> <bytes>\r\n<data> \r\n

delete <key>

decr <key>

<value>\r\n

**Advanced Features** CAS (Check and Set)

gets <key>\r\n

cas <key> <flags> <exptime>

<bytes> <cas unique>\r\n<data

>\r\n

Example:

conditions.

cas mykey 0 3600 5 12345\r\nvalue\r\n

Flags are an arbitrary 16-bit integer that the client stores along with the data. It is opaque to the server.

Clients can use flags to indicate the type of data being stored (e.g., serialized object, compressed data).

When data is retrieved via get, the flags are also returned.

## Flags

# Deletes data associated with the specified key. Time <time>\r\n is an optional delay before deletion (in seconds). delete mykey\r\n incr <key> Increments the value of the key by value. The value <value>\r\n must be an integer. Example:

incr counter 1\r\n

decr counter 1\r\n

Retrieves data with a unique

Stores data only if the CAS

current value. Prevents race

identifier matches the

must be an integer.

Example:

CAS identifier.

gets mykey\r\n

Example:

# Multi-Get

get <key1> <key2> ... <keyN>\r\n

Retrieves multiple keys in a single request, reducing network overhead.

Example:

get key1 key2 key3\r\n

## Expiration

Expiration time is specified in seconds. A value of O means the item never expires (although it may be evicted from the cache if memory is needed).

If the expiration time is greater than 30 days (2592000 seconds), it is treated as a Unix timestamp.

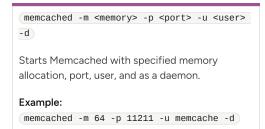
## LRU (Least Recently Used)

Memcached uses an LRU algorithm to evict items from the cache when it runs out of memory. Least recently used items are removed first.

Page 1 of 2 https://cheatsheetshero.com

# **Configuration & Management**

## Starting Memcached



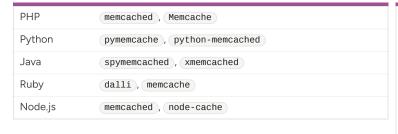
## **Configuration Options**

| -m<br><memory<br>&gt;</memory<br>  | Sets the maximum memory to use for cache items, in MB.                   |
|--|--|
| -p<br><port></port>  | Sets the port number to listen on (default: 11211).                      |
| -u<br><user></user>  | Runs the daemon as the specified user.                                   |
| -d   | Runs Memcached as a daemon.  |
| -1   |  |
| <ip_addr< td=""><td>Bind Memcached to a specific IP address. Useful for multi-homed servers.</td></ip_addr<> | Bind Memcached to a specific IP address. Useful for multi-homed servers. |

## Stats

## **Client Libraries**

# Popular Libraries



# Basic Usage (Python - pymemcache)

```
from pymemcache.client.base import Client

client = Client('127.0.0.1:11211')

client.set('my_key', 'my_value')

value = client.get('my_key')

print(value)
```