Lesson: Redis & Memcached

What you'll learn

- Why caching exists and how an in-memory cache speeds up an app.
- Redis core data types and common use cases (caching, sessions, queues).
- What Memcached is and how it differs.
- The trade-offs between Redis and Memcached so you can pick one.
- Basic install and hands-on commands for both.

By the end you can explain why a cache is in a stack, install Redis/Memcached, and run basic commands at a recognise-and-operate level.

The lesson

1. Why cache at all?

Reading from a database or computing a result can be slow — disk access, JOINs, network round-trips. A **cache** keeps a copy of frequently used data in **RAM** (main memory), which is dramatically faster than disk. The app checks the cache first; only on a **miss** does it hit the slower database, then stores the answer in the cache for next time.

```
Without cache                       With cache
-------------                       ----------
app --> [ database (disk) ]         app --> [ cache (RAM) ] --hit--> fast answer
         slow every time                              |
                                                      +--miss--> [ database ] --> store in cache
```

This is the **cache-aside** pattern: look in cache → if missing, read DB and populate cache. Because RAM is finite, caches use **expiry (TTL — time to live)** and **eviction** (drop old/least-used entries when full). A cache is meant to be **disposable**: if it's wiped, the app still works, just slower while it refills.

The two you'll meet are **Redis** and **Memcached**. Both are in-memory **key-value** stores: you store a value under a key and fetch it back by that key.

2. Redis basics

Redis (REmote DIctionary Server) is an in-memory data store that is far more than a simple cache — its values can be rich data structures. Install and connect:

```
sudo apt update
sudo apt install -y redis-server
sudo systemctl enable --now redis-server
redis-cli ping            # -> PONG means it's alive
```

`redis-cli` is the interactive client. Default port is `6379`.

3. Redis data types and commands

The power of Redis is its **data types** — the value isn't just a string:

- **String** (also used for numbers):

  ```
  SET user:1:name "Layla"
  GET user:1:name
  SET visits 0
  INCR visits            # atomic increment -> 1
  EXPIRE user:1:name 60  # auto-delete after 60 seconds (TTL)
  TTL user:1:name        # seconds left
  ```

- **Hash** (a map of fields → values, good for objects):

  ```
  HSET user:1 name "Layla" role "editor"
  HGETALL user:1
  ```

- **List** (ordered, good for queues):

  ```
  LPUSH jobs "send-email"      # push to head
  RPOP jobs                    # pop from tail -> FIFO queue
  ```

- **Set** (unique members) and **Sorted Set** (members with a score, great for leaderboards):

  ```
  SADD tags "linux" "nginx"
  ZADD scores 100 "omar" 80 "sara"
  ZREVRANGE scores 0 2 WITHSCORES   # top players
  ```

A handy mental note: keys are commonly namespaced with colons like `user:1:name` — that's a convention, not a requirement.

4. Redis use cases

- **Caching** — the cache-aside pattern from section 1; store DB results with a TTL.
- **Sessions** — web apps store a user's login session under a key like `session:<id>` with a TTL, so any app server can read it. Our lab's app stacks use Redis here.
- **Queues / background jobs** — `LPUSH` work onto a list, workers `RPOP` and process it.
- **Rate limiting & counters** — `INCR` + `EXPIRE` to count requests per minute.
- **Leaderboards / rankings** — sorted sets.
- **Pub/Sub** — lightweight publish/subscribe messaging.

Redis can also **persist** to disk (snapshots called RDB, or an append-only log called AOF), so it can survive a restart — useful when it holds sessions or a queue you can't afford to lose.

5. Memcached basics

Memcached is a deliberately minimal, multi-threaded, in-memory **key-value cache**. Values are just strings/blobs — no rich types. Install and poke at it:

```
sudo apt install -y memcached
sudo systemctl enable --now memcached
# quick test over the text protocol:
printf 'set foo 0 60 3\r\nbar\r\nget foo\r\n' | nc -q1 127.0.0.1 11211
```

Default port is `11211`. The command above sets key `foo` to `bar` with a 60-second TTL, then reads it back. Memcached does one thing — fast, volatile caching — and does it simply.

6. Redis vs Memcached — trade-offs

Both are fast in-memory key-value stores. Choose based on what you need:

Aspect	Redis	Memcached
Data types	Rich: strings, hashes, lists, sets, sorted sets	Strings/blobs only
Persistence	Optional (RDB/AOF) — can survive restart	None — purely volatile
Threading	Mostly single-threaded core	Multi-threaded
Features	Pub/Sub, TTL, scripting, transactions, queues	Just get/set with TTL
Best for	Sessions, queues, counters, anything beyond plain caching	Simple, very high-throughput object caching

Rule of thumb: **default to Redis** — it does everything Memcached does plus much more, and it's what most modern stacks use. Reach for Memcached only when you specifically want a bare, multi-threaded cache and nothing else.

7. Operating notes

- Service status: `systemctl status redis-server` / `systemctl status memcached`.
- Inspect Redis live: `redis-cli`, then `INFO`, `DBSIZE`, `KEYS *` (avoid `KEYS *` on big production instances — use `SCAN`).
- Secure them: bind to internal IPs only and set a password (`requirepass` in `/etc/redis/redis.conf`). Never expose `6379`/`11211` to the open internet.
- Remember a cache is disposable: design the app to refill it on a miss, not to depend on it as the source of truth.

Dig deeper

- Redis documentation (official)
- Redis data types intro
- Memcached wiki (official)
- DigitalOcean: How To Install and Secure Redis on Ubuntu
- AWS: Redis vs Memcached comparison

Search terms

- `why use caching in web applications`
- `redis data types tutorial strings hashes lists`
- `redis session store explained`
- `redis vs memcached when to use which`
- `redis-cli basic commands SET GET EXPIRE`
- `cache-aside pattern explained`

Check yourself

1. Why is reading from an in-memory cache faster than from a database, and what does it mean that a cache is "disposable"?
2. Describe the cache-aside pattern in your own words.
3. Name three Redis data types and one use case for each.
4. Give two things Redis can do that Memcached cannot.
5. What is a TTL, and which Redis command sets one on an existing key?

Revision #3
Created 2026-05-19 18:09:00 UTC by Eslam Shapsough
Updated 2026-05-19 18:13:00 UTC by Eslam Shapsough