# Key-Value Stores

Sometimes rDBs are too cumbersome ~~@~~ ~~t~~ : when weighed against the services they offer. Also, they impose a tabular struc. on data viz not reqd. everytime. In those cases, we use non-rDBs for more flexibility and ease

Key-Value store → i) A popular non-rDb (i.e. noSQL)
ii) Data stored as k-v pairs.
iii) Allows mapping of keys (~~strings~~ usually strings) to ~~values~~ arbitrary values.

eg1:

| Key | Value |
|---|---|
| foo | -48.8 |
| bleh | ApplePie |
| whut | [x, y, z] |

eg 2: hashTables

k-v store →
- flexible → not rigorous as no imposed struc.
- simple → nothin simpler to store data compared to a k-v map.
- fast

Use:

1) Caching : Values → Responses to network reqs.
Keys → Hash, IP address, username

2) Dynamic config : Basically just storing a parameter that diff. parts of your sys. rely on in a separate place so all these parts can access it

eg :- using config" when you wanna switch what code you wanna run depending on what envirnt. you're in → (development, prod", testing)

Here, this param. is stored in a k-v store (usually an obj. in .json or .yaml file) as working with diff. values of this same param (key) is a lot easier

NOTE: Advantage of using a k-v store → ∵ values are accessed directly thru keys, no search thru DB or any other fancy lookup reqd.
∴ Latency ↓, Throughput ↑ of our sys.

eg:- DynamoDB, Redis, Etcd, Zookeeper

NOTE: 2 types of k-v stores based on where they write their data to:

i) Write data to disc → a) Slow to get back data from disc
b) Data persists even after k-v store server itself crashes

ii) Write data to mem → a) Faster to retrieve data from mem
b) Data lost if k-v store server crashes

Use :- Caching → As honestly, it doesn't affect the sys. much if caches crash but we do want reading from cache = fast

Date:
M T W T F S S

~~Also, in~~

∴ when a caching involving k-v store goes down, we pretty much don't lose anything except some cache hits viz not that imp. a loss.

---

eg' of k-v stores:

1) Etcd
   - → strong consistency
   - → high availability
   - → Use: implement leader elec$^n$ in a sys.
   - → writes data to disc

2) Redis
   - → writes data to mem. a.k.a in mem. k-v store
   - → V. lil persistent storage
   - → Uses: best-effort, fast caching implementa$^n$ of 'rate limiting'

3) Zookeeper
   - → writes data to disc
   - → strong consistency
   - → high availability
   - → Use: Store imp config$^n$ Implement leader elec$^n$