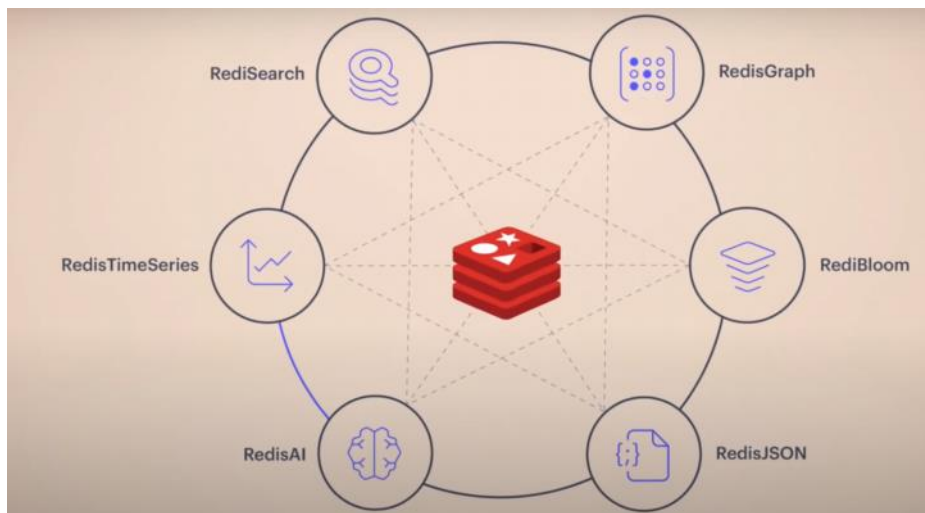
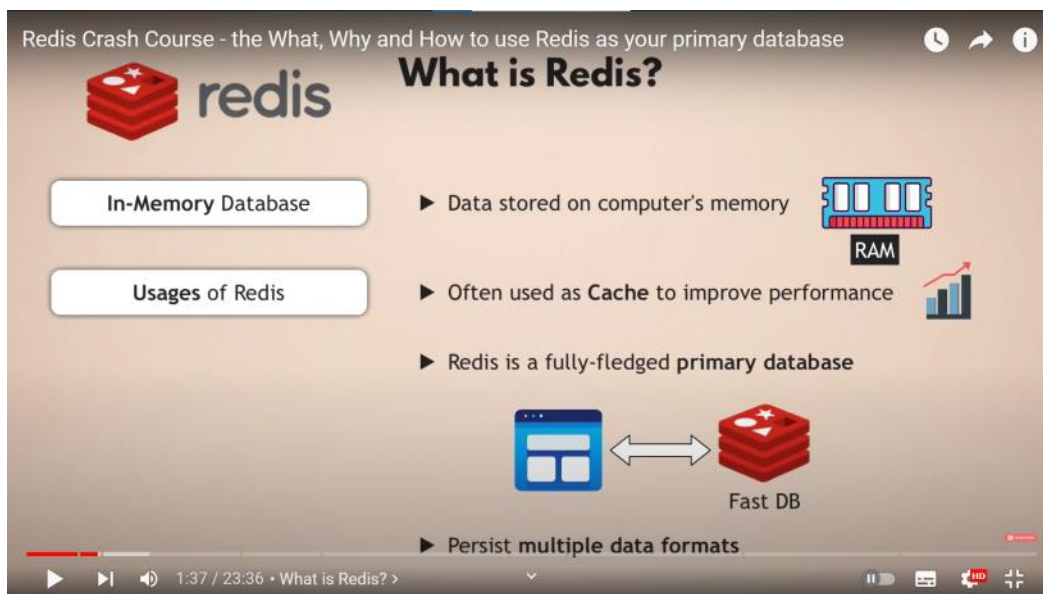


Reddis

Thursday, October 12, 2023 4:45 PM

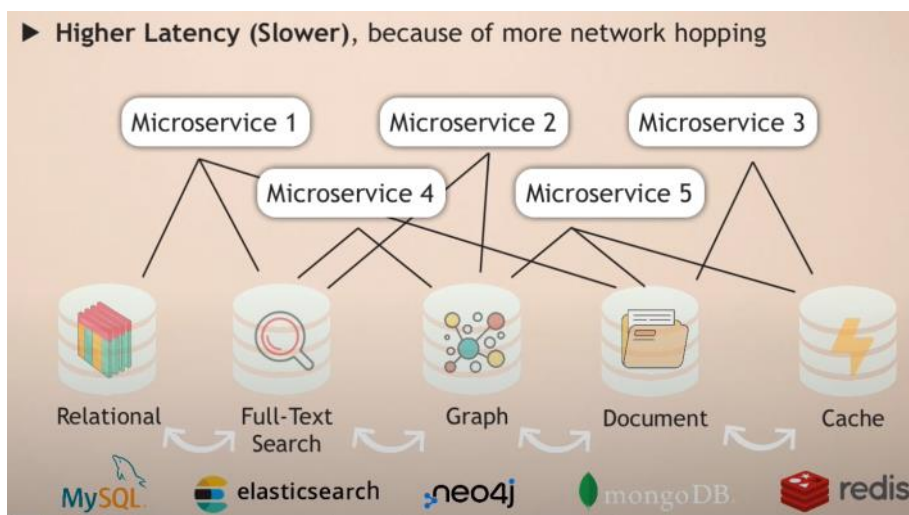


Redis Crash Course - the What, Why and How to use Redis as your primary database

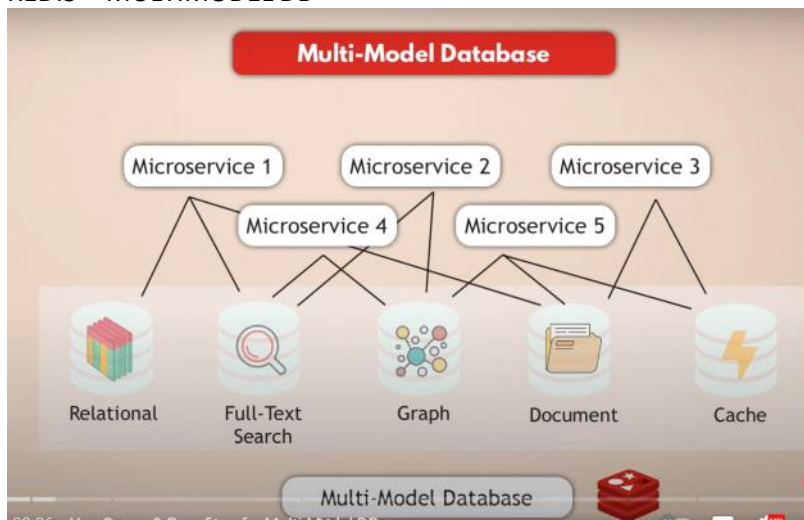
Challenges of multiple data services

- Data services need to be deployed and maintained
- Know-How needed for each data service
- Different Scaling & Infrastructure Requirements

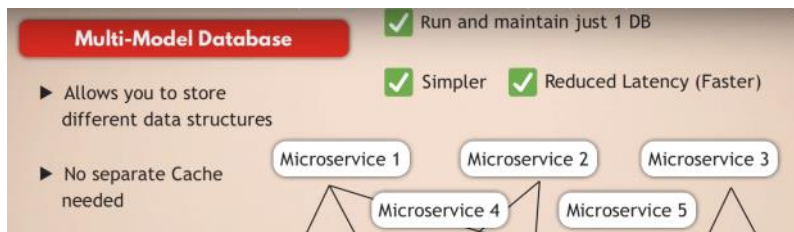
3:21 / 23:36 • Use Cases & Benefits of a Multi-Model DB >



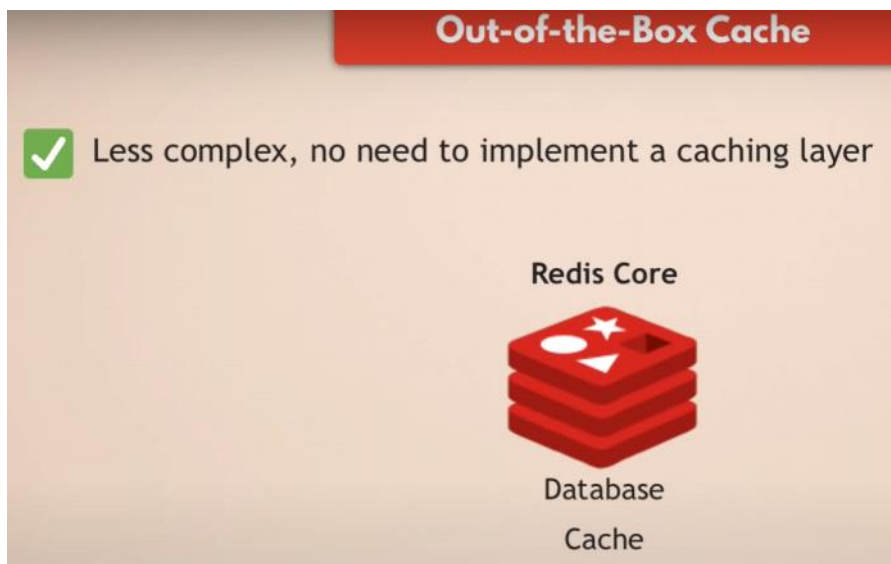
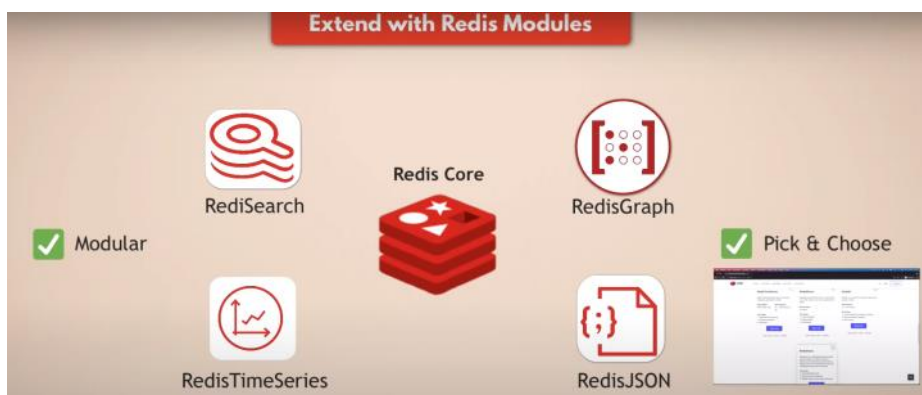
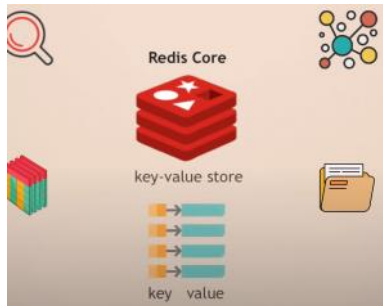
REDIS==MULTIMODEL DB

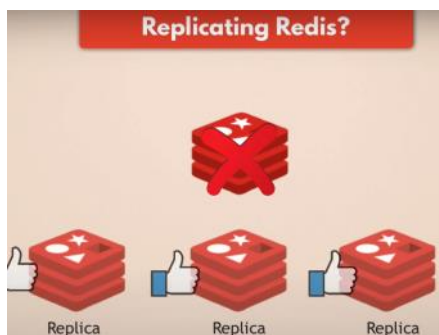
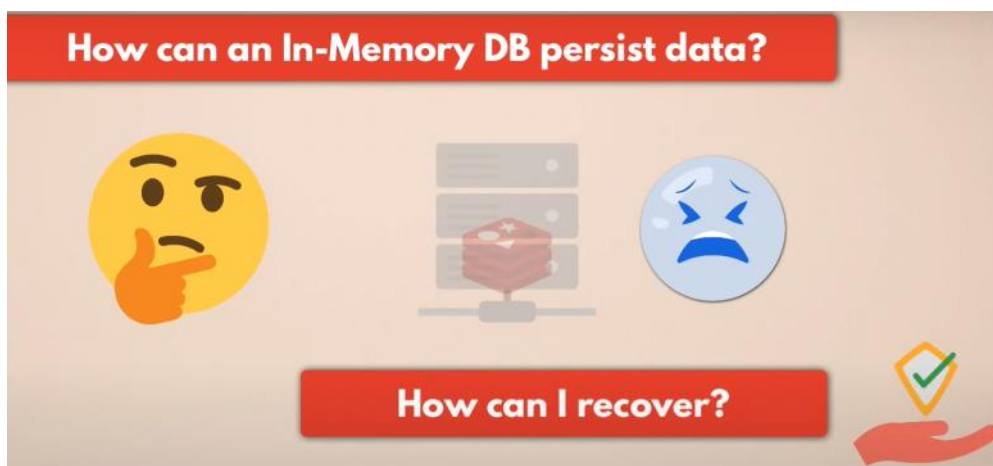
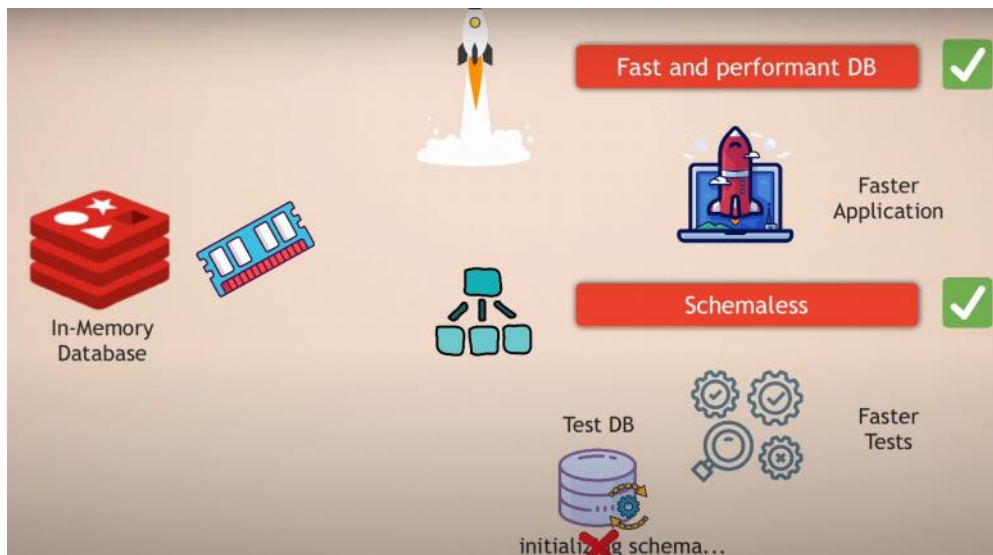


BENEFITS OF USING REDIS:



HOW REDIS WORKS?






But what if all replica goes down?, how to persist then?

Redis Crash Course - the What, Why and How to use Redis as your primary database

So, how can we keep the data safely **persisted**?


1) Snapshotting (RDB)

- ▶ Produces single-file point-in-time snapshots of your dataset
- ✓ Great for backups & disaster recovery
- ✗ You may lose the latest minutes of data



dump.rdb

2) Append Only File (AOF)

- ▶ Logs every write operation continuously
- ▶ When restarting Redis, it will re-play the AOF to rebuild the state
- ✓ Much more durable
- ✗ Can be slower than RDB







appendonly.aof

8:35 / 23:36 • Data Persistence & Durability with Redis (Snapshotting and AOF) >



 Best: Use **both** persistence options

1) Snapshotting (RDB) + **2) Append Only File (AOF)**


- ▶ Regular snapshots for DB backups
- ▶ Persisting all operations one after the other

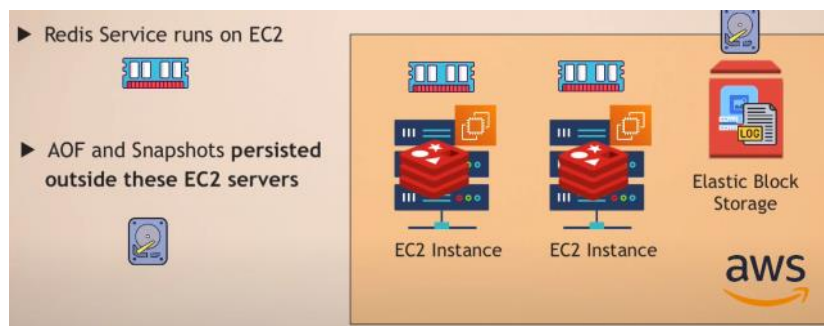
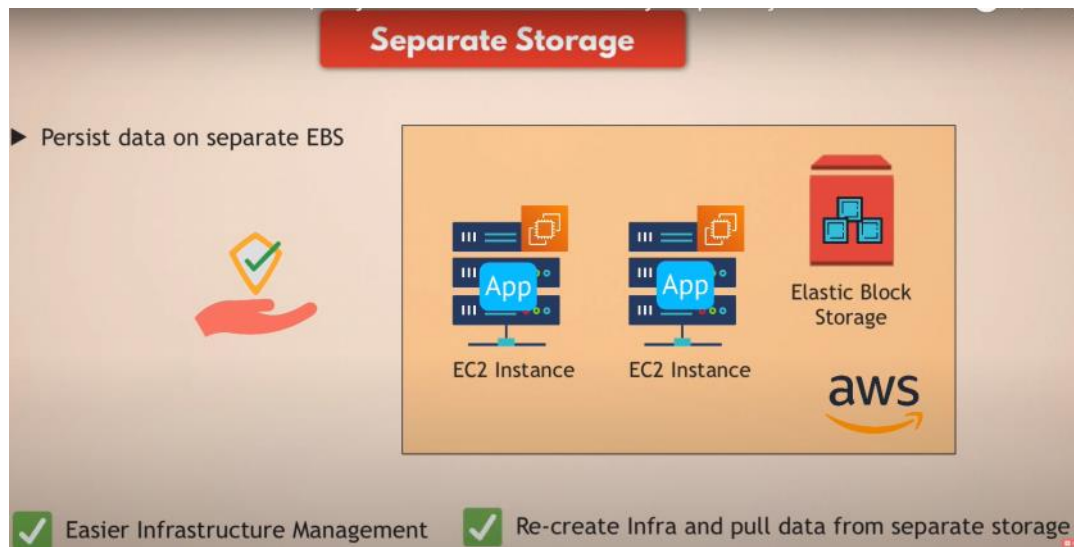
    

Where are these persistence files stored?

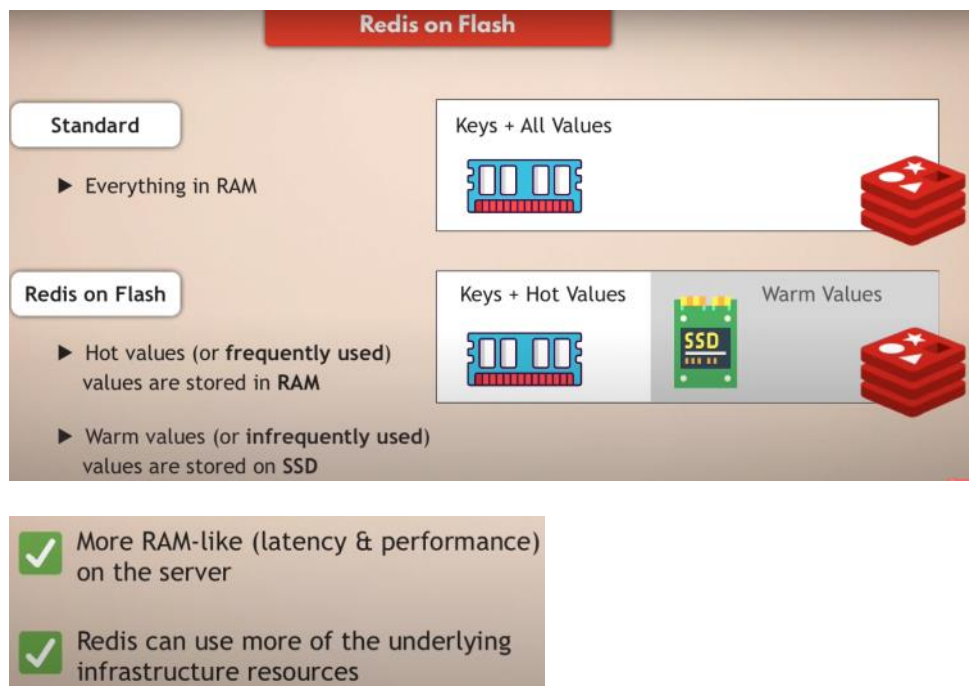
 

Server, where DB service runs Server, where your data is backed up

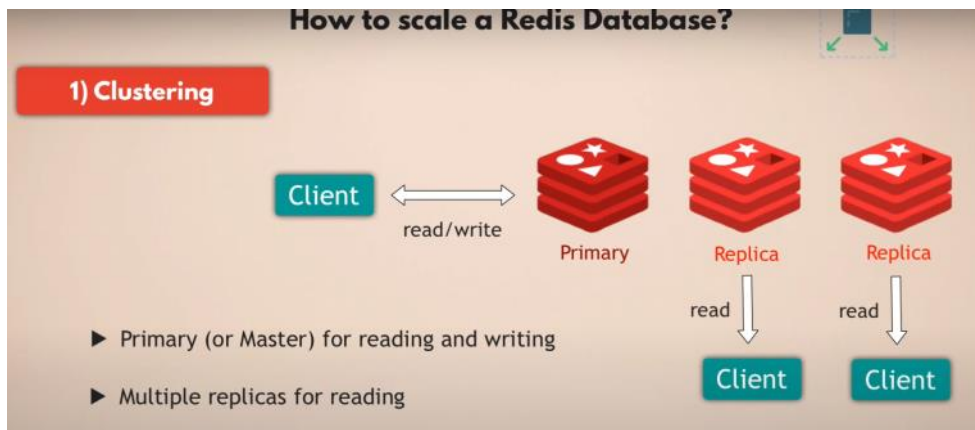
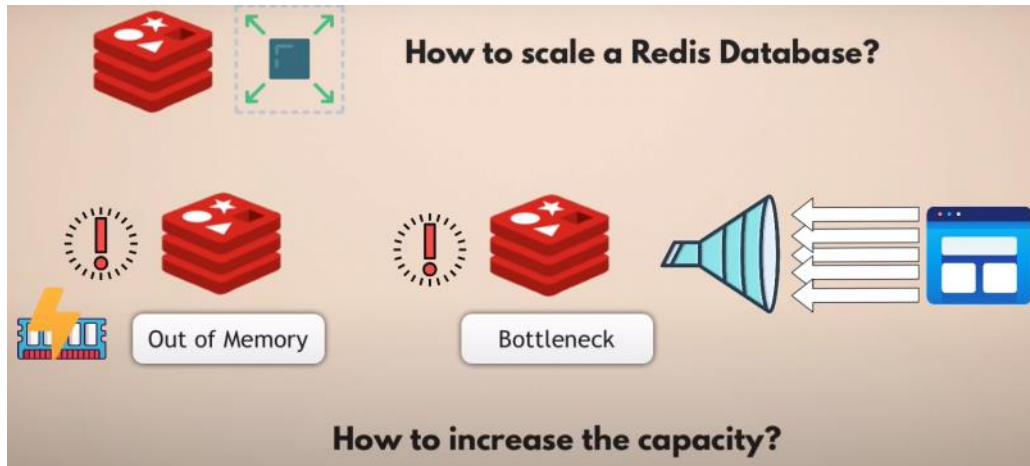
 Best Practice: **Separate** Persistent Storage from Data Service



OPTIMISING COST?:



✓ Lower Infrastructure Costs



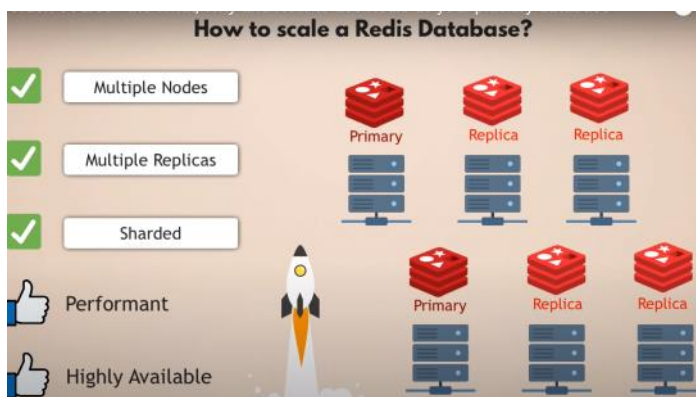
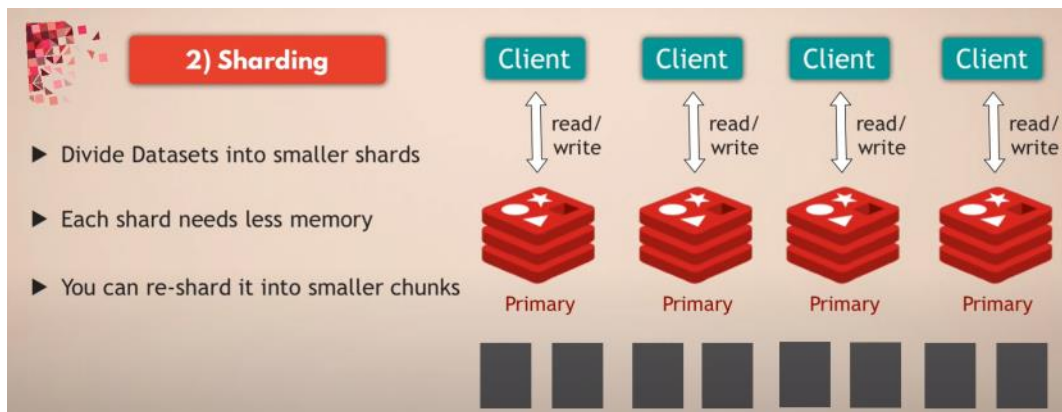
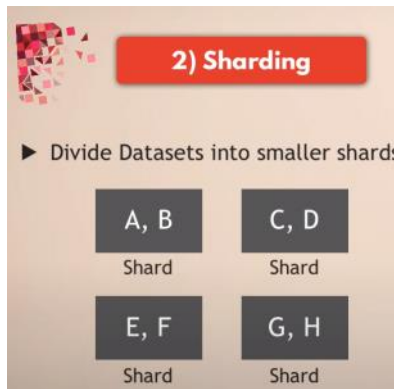
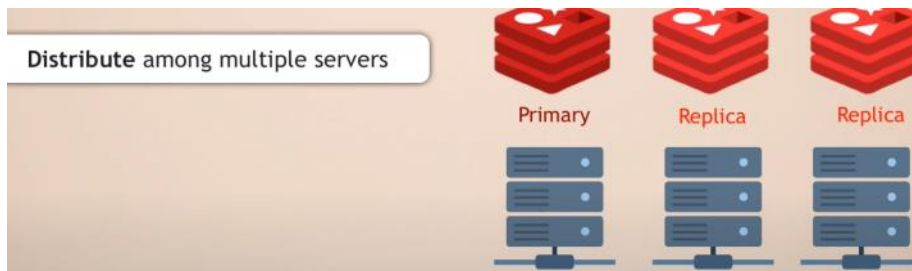
✓ Handle more requests

✓ High Availability

Don't keep replica's on same server of primary Instance:



Distribute among multiple servers



HIGH AVAILABILITY AND PERFORMANCE IN VARIOUS REGIONS:

Use Cases

Users geographically distributed

- ▶ We want to distribute our data service close to the users

Disaster Recovery

- ▶ Switch over to another data center, when one goes down

- ✓ Replicas of Redis Cluster in different regions
- ✓ Data should be replicated to all clusters
- ✓ Each cluster should be able to accept reads/writes

Soln :

Local Cluster of Europe region **Local Cluster of US region**

- ▶ Each Redis cluster acts as local instances in each region
- ▶ Syncers contact remote masters for replication

- ✓ Lower Latency
- ✓ Disaster Recovery

Active-Active Geo Distribution

ENTERPRISE

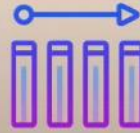
- ▶ Redis cluster can update data still independently
- ▶ Sync data when connection is re-established

► SET key1 "a" ⚡ ► SET key1 "b"



How does Redis **resolve** changes to the **same dataset**?

How does Redis **ensure data consistency**?



CRDT - Conflict-Free Replicated Data Types



Outcome of concurrent writes is **predictable** & based on a set of rules



Dataset will eventually **converge** to a single, consistent state

Active-Active Geo Distribution




All parallel changes are intelligently resolved

Redis with Kubernetes





Running Redis in Kubernetes




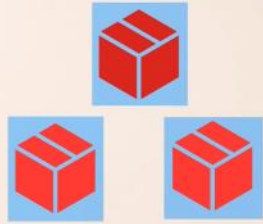
Self-Managed

- ▶ Deploy Redis as Helm Chart or K8s manifest files





- ▶ Same Replication and Scaling







redis cluster



- ▶ Hosts are K8s Pods instead of virtual or physical server



Managed Redis Cluster




Managed Redis Cluster


- ▶ Deploy Redis Enterprise as K8s Operator

Operator

- ▶ Bundle all resources to operate a certain service



"software"
operator



mysql-operator

- ▶ How to create the db cluster
- ▶ How to run it
- ▶ How to synchronize the data
- ▶ How to update

