Redis stores data in memory, it means it stores it in ram that is fast, used for cache data frequently used data.

Ram --> Nano seconds, Hard Disk --> MiliSeconds

Redis cloud connection:

Application.yaml

spring:  
 application:  
 name: journal  
 data:  
 redis:  
 host: redis-11774.crce178.ap-east-1-1.ec2.redns.redis-cloud.com  
 port: 11774  
 password: mW0bqCJ1WAMGNhVbrS4o0wSHrJDVw6lz

**REDIS SERVICE:**

import com.fasterxml.jackson.databind.DeserializationFeature;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.redis.core.RedisTemplate;  
import org.springframework.stereotype.Service;  
  
import java.util.concurrent.TimeUnit;  
  
@Slf4j // autmatically injects logger object  
@Service  
public class RedisService {  
  
 // In-Memory location  
 // Allocated in RAM  
 // KEY-VALUE pairs  
 // Used to set values and retrieve values  
 @Autowired  
 private RedisTemplate redisTemplate;  
  
  
 public ObjectMapper giveObjectMapper() {  
 return new ObjectMapper();  
 }  
  
 // <T> tells Java: "Hey, this method can return any type, and we’ll determine what it is when calling the method."  
 // <T> is a class, second T is for object that will be returned actually  
  
 public <T> T get(String key, Class<T> entityClass) {  
 try {  
 Object obtained = redisTemplate.opsForValue().get(key); // Directly cast to String  
 if (obtained != null) {  
 ObjectMapper mapper = giveObjectMapper() ;  
 return mapper.readValue(obtained.toString(), entityClass); // Deserialize correctly  
 }  
 } catch (Exception e) {  
 *log*.error("Error retrieving key '{}' from Redis: {}", key, e.getMessage(), e);  
 }  
 return null; // Return null only if not found or on failure  
 }  
  
  
 public void set(String key, Object setting, Long tDuration){  
 try{  
  
 ObjectMapper mapper = giveObjectMapper();  
 String jsonResponse = mapper.writeValueAsString(setting);  
 redisTemplate.opsForValue().set(key,jsonResponse,tDuration, TimeUnit.*SECONDS*);  
  
 }  
 catch (Exception e) {  
 *log*.error("Error occured -> " , e);  
 }  
  
 }  
  
}