# Redis

Data Engineering


redis

# What is Redis?

Remote Dictionary Server

The “remote” in the name refers to the fact that Redis is deployed as a server application that runs separately from the client applications that access it

# What is Redis?

- Redis is an open source.
- It is in-memory data structure store.
- It is used as a database, cache, and message broker and queue.
- The Key-Value Data Model.
- Redis as a Cache entirely in the memory.
- Commands and Pipelining.
- Partitioning with Redis Cluster.
- Publish/Subscribe Messaging.

# Why Redis?

# Reduce indexing overhead

**Pandas Index**

```
import pandas as pd
data = {'username': ['user1’,  'user2', 'user3'],
        'age': [25, 30, 35],
        'email': ['user1@example.com', 'user2@example.com’,
        'user3@example.com']}
df = pd.DataFrame(data)

# Create an index on the 'username' column
df.set_index('username', inplace=True)
```

**key-value pairs**

```
data = {'user1': {'age': 25, 'email': 'user1@example.com'},
        'user2': {'age': 30, 'email': 'user2@example.com'},
        'user3': {'age': 35, 'email': 'user3@example.com'}}
s = pd.Series(data)

# Access values directly using keys (index)
print(s['user1'])  # Output: ?
```

Redis minimizes the performance impact and resource utilization associated with indexing

# No Schema Overhead

- Each key-value pair in Redis can have a different structure, containing only the attributes relevant to that user.

- For example, one user1 might include fields like "username", "email", and "age", while user2 might include fields like "username", "email", "age", and "address".

## Redis pipeline And why is it useful?

In Redis, a pipeline is a technique used to improve the performance of client-server communication. It allows a client to send multiple commands to the server without waiting for the response to each command before sending the next one.

1. To improve performance Optimization

```
Redis-cli MULTI
Redis-cli SET key1  53
Redis-cli SET key2 “string2”
Redis-cli SET key3 “string3”
Redis-cli EXEC
```

2. Atomicity and Consistency: all three increments are executed as a single atomic unit

```
Redis-cli MULTI
Redis-cli INCR counter
Redis-cli INCR counter
Redis-cli INCR counter
Redis-cli EXEC
```

3. Easy for Debugging

## Publisher/Subscriber Message

It is a key feature of Redis that enables asynchronous communication between different components of a distributed system.

- Publisher: The component that produces messages and sends them to the message broker.
- Subscriber: The component that receives and processes the messages sent by the publisher.
- Message Broker: The intermediary component that receives messages from publishers and distributes them to the appropriate subscribers.

## Some key benefits

- Publishers and subscribers are decoupled, they don't have to know each other's existence
- Redis can handle a large number of publishers and subscribers without affecting the overall system performance
- Publishers and subscribers can operate independently
- New publishers and subscribers can be added to the system without affecting the existing components.

# Redis Data Types

- **strings** — a sequence of binary safe bytes up to 512 MB
- **hashes** — a collection of key value pairs
- **lists** — an in-insertion-order collection of strings
- **sets** — a collection of unique strings with no ordering
- **sorted sets** — a collection of unique strings ordered by user defined scoring
- **Bitmaps** – a data type that offers bit level operations
- **HyperLogLogs** – a probabilistic data structure to estimate the unique items in a data set

# Radis Strings

<u>Commands associated with strings:</u>

- SET: sets a value to a key
- GET: gets a value from a key
- DEL: deletes a key and its value
- INCR: atomically increments a key
- INCRBY: increments a key by a designated values
- EXPIRE: the length of time that a key should exist (denoted in seconds)

# Hashes

•**hashes** — a hash can contain multiple field-value pairs, where each field is a unique identifier within the hash, and each value is associated with a field.

•a hash can represent a user in a database, with each field representing a different attribute of the user (e.g., username, email, age), and each value storing the corresponding data for that attribute. redis-cli

# Hashes

HSET user:123 name "John Smith" email "john@example.com" age 35
HGET user:123 name age

HKEYS user:123
HVALS user:123
HLEN user: 123

# Lists

- **lists** — an in-insertion-order collection of strings

```
# Push elements to the beginning of the list
LPUSH recent_searches "cats" "dogs" "birds"

# Push elements to the end of the list
RPUSH recent_searches "fish" "reptiles"

# Get the length of the list
LLEN recent_searches  # Returns 5

# Get a range of elements from the list
LRANGE recent_searches 0 2  # Returns ["birds", "dogs", "cats"]

# Pop an element from the beginning of the list
LPOP recent_searches  # Returns "birds"

# Pop an element from the end of the list
RPOP recent_searches  # Returns "reptiles"
```

# Sorted Sets

- **sorted sets** — a collection of unique strings ordered by user defined scoring

```
# Add elements to a set
SADD product_tags "electronics" "computers" "laptops"
SADD product_tags "smartphones" "tablets"

# Check if an element is a member of the set
SISMEMBER product_tags "laptops"  # Returns 1 (true)
SISMEMBER product_tags "televisions"  # Returns 0 (false)

# Get all the elements in the set
SMEMBERS product_tags  # Returns ["computers", "electronics", "laptops", "smartphones", "tablets"]

# Perform set operations
SUNION product_tags another_product_tags  # Perform a union operation
SINTER product_tags another_product_tags  # Perform an intersection operation
SDIFF product_tags another_product_tags  # Perform a difference operation
```

# Bitmaps

- **Bitmaps** – a data type that offers bit level operations

```
# Set a bit at a specific offset
SETBIT user_activity 123 1  # Set the bit at offset 123 to 1 (active)

# Get the bit value at a specific offset
GETBIT user_activity 123     # Returns 1, indicating the user was active

# Count the number of active users
BITCOUNT user_activity       # Returns the number of active users

# Perform a bitwise operation
BITOP AND active_users user_activity_1 user_activity_2
```

# HyperLogLogs

- **HyperLogLogs –** a probabilistic data structure to estimate the unique items in a data set

```
# Add elements to the HyperLogLog
PFADD unique_visitors "user1" "user2" "user3" "user4" "user5"

# Get the approximate number of unique elements
PFCOUNT unique_visitors  # Returns 5

# Merge multiple HyperLogLogs
PFADD unique_visitors_2 "user3" "user4" "user5" "user6" "user7"
PFMERGE unique_visitors_merged unique_visitors unique_visitors_2
PFCOUNT unique_visitors_merged  # Returns an approximate count of 7 unique visitors
```

# Hash Data Example

```
Key: "accountID:5531"
Value: "name" => “Majed Al-Ghandour"
       "username" => “cool_Dr_Al"
       "country" => "USA"
       "lastLogin" => "1383147407"
       "loginCount" => "3691"
       "lastPaymentSync" => "1383256813"
       "signup" => "1381254812"
       "isAdmin" => "false" "
```

# Business Case

**Gaming Industry**: In the gaming industry, Redis is used for real-time leaderboards, player sessions, matchmaking, and in-game messaging systems. Its ability to handle high concurrency and low latency makes it well-suited for real-time gaming applications.

# Business Case

**E-commerce Platforms**: In e-commerce platforms, Redis is often used for caching product catalogs, session storage for user sessions, and managing shopping cart data. Its fast read and write operations help improve website performance and provide a better user experience.

# Business Case

**Social Media Platforms**: Social media platforms use Redis for caching user profiles, timelines, and activity feeds. Redis's ability to handle high write throughput and provide sub-millisecond latency is crucial for delivering real-time updates to users.

# Business Case

**Financial Services**: In the financial services sector, Redis is used for caching market data, processing transactions, and managing real-time risk analysis. Its high availability and clustering capabilities ensure data integrity and reliability in critical financial systems.

# Business Case

**Content Delivery Networks (CDNs)**: CDNs use Redis for caching static content, session storage, and managing distributed systems. Its support for data partitioning and replication ensures fast and reliable content delivery to users worldwide.

# Redis Installation on Mac

Open terminal:

```
brew --version
brew install redis

# to start the server:
redis-server

Or brew services start redis

#to stop it
Ctlr + C
brew services stop redis
brew services restart redis
```

Docs (redis.io)

## SET and GET

Connect to Redis:

```
Redis-cli
127.0.0.1:6379>SET mykey “ABC”
OK
127.0.0.1:6379 >GET mykey
“abc”
```

## LPUSH

```
# Initial list creation and insertion
127.0.0.1:6379> LPUSH myqueue "first-1"
# Output: (integer) 1

# Add more elements to the front of the list
127.0.0.1:6379> LPUSH myqueue "second-2" "third-3"
# Output: (integer) 3

# Retrieve all elements of the list
127.0.0.1:6379> LRANGE myqueue 0 -1
# Output:
# 1) "third-3"
# 2) "second-2"
# 3) "first-1"
```

## LPUSH

```
127.0.0.1:6379> LPUSH mylist "one" "two" "three"
"four”

127.0.0.1:6379>LRNAGE mylist 0 0 #the first element
127.0.0.1:6379>LRNAGE mylist 1 1 #the second element
127.0.0.1:6379>LRNAGE mylist -1 -1 #the last element
```

# INCR

```
127.0.0.1:6379> INCR mycounter
127.0.0.1:6379> INCR mycounter
127.0.0.1:6379> INCR mycounter

127.0.0.1:6379> DECR mycounter
127.0.0.1:6379> DECR mycounter

127.0.0.1:6379> 5 INCR mycounter
```

What is the output?

## Interactive Mode

```
127.0.0.1:6379> PING
PONG
```

### Pub/sub mode

```
127.0.0.1:6379>psubscribe "*" #to match any channel names

Open another terminal
Redis-cli publish FirstChannel Channel-1
Redis-cli publish SecondChannel Channel-2
```

# Demo1

lpush DataEngineering redis
lpush DataEngineering MongoDB
lpush DataEngineering Cassandra
lpush DataEngineering Hadoop
RPUSH DataEngineering SQL

C:\Redis-x64-3.2.100\redis-cli.exe

```
127.0.0.1:6379> lpush DataEngineering redis
(integer) 1
127.0.0.1:6379> lpush DataEngineering MongoDB
(integer) 2
127.0.0.1:6379> lpush DataEngineering Cassandra
(integer) 3
127.0.0.1:6379> lpush DataEngineering Hadoop
(integer) 4
127.0.0.1:6379> RPUSH DataEngineering SQL
(integer) 5
127.0.0.1:6379>
```

# Demo 2

C:\Redis-x64-3.2.100\redis-cli.exe

```
127.0.0.1:6379> sadd DataScience redis
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379> sadd DataScience mongodb
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379> sadd DataScience hadoop2
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379> sadd DataScience MySQL
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379> smembers DataScience
1) "hadoop2"
2) "MySQL"
3) "redis"
4) "mongodb"
127.0.0.1:6379>
```

sadd DataScience redis
sadd DataScience mongodb
sadd DataScience hadoop2
sadd DataScience MySQL
smembers DataScience

# Redis Enterprise Cloud

https://redislabs.com/redis-enterprise-cloud/

Log In | UMGC | Downloads | Try Redis Enterprise | Redis Labs | Redis Enterprise Cloud – Fully M

redislabs.com/redis-enterprise-cloud/overview/

Apps | Azure Lab Services | Colab | triveratech | Other bookmarks

redislabs HOME OF REDIS

CLOUD SOFTWARE PLATFORM SOLUTIONS LEARN

Login Try Free

Overview Cloud Migration Cloud-native Pricing

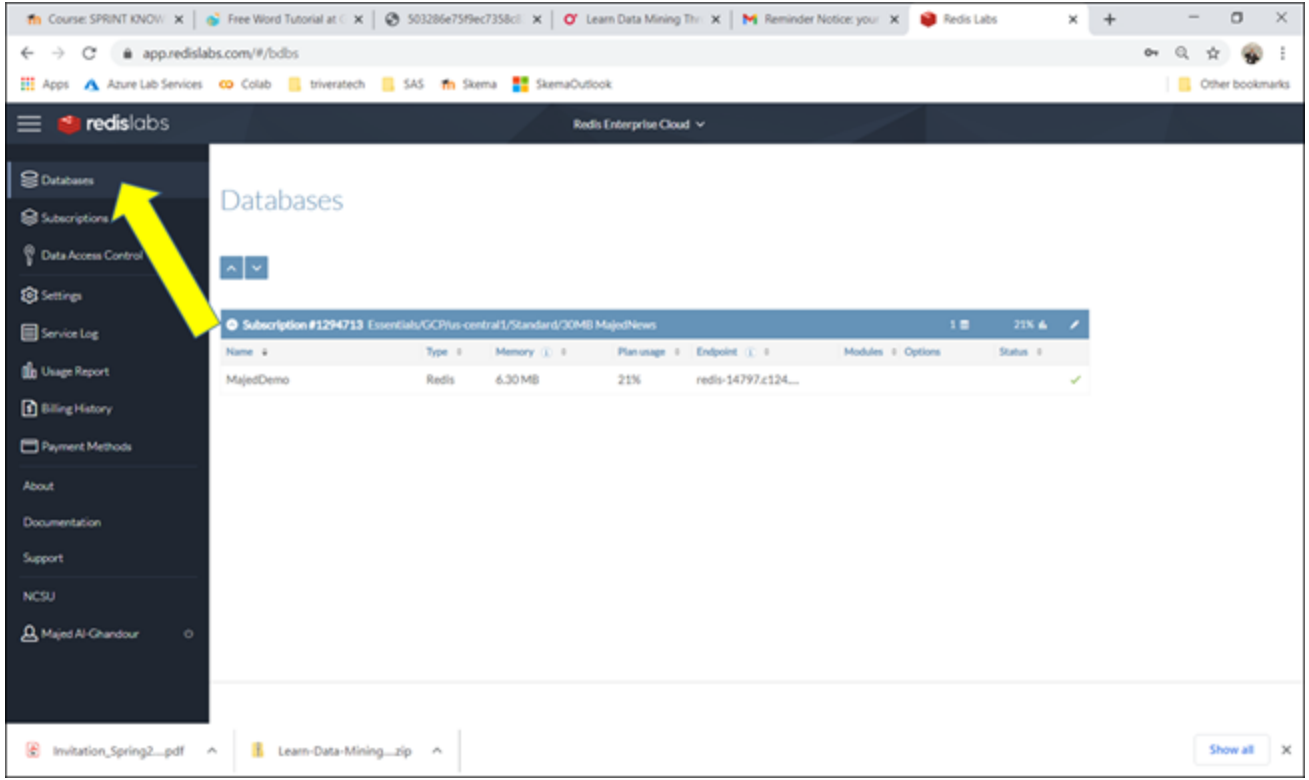
# Redis Enterprise Cloud

## Fully managed cloud service by the makers of Redis

Start Free

Ready to use a simple, easy to deploy database to improve your team's performance?

# Demo 3


Redis Labs
app.redislabs.com/#/bdbs
Apps
Azure Lab Services
Colab
triveratech
SAS
Skema
SkemaOutlook
Other bookmarks
redislabs
Redis Enterprise Cloud
Databases
Subscriptions
Data Access Control
Settings
Service Log
Usage Report
Billing History
Payment Methods
About
Documentation
Support
NCSU
Majed Al-Ghandour
Databases
Subscription #1294713 Essentials/GCP/us-central1/Standard/30MB MajedNews
Name
Type
Memory
Plan usage
Endpoint
Modules
Options
Status
MajedDemo
Redis
6.30 MB
21%
redis-14797.c124...
Invitation_Spring2....pdf
Learn-Data-Mining....zip
Show all

# Demo 4

Select C:\Users\mnalghandour\Downloads\Redis-x64-3.2.100\redis-server.exe

```
[8032] 25 Jun 22:33:15.395 # Warning: no config file specified, using the default config. In order to specify a config f
ile use C:\Users\mnalghandour\Downloads\Redis-x64-3.2.100\redis-server.exe /path/to/redis.conf

                Redis 3.2.100 (00000000/0) 64 bit

                Running in standalone mode
                Port: 6379
                PID: 8032

                      http://redis.io

[8032] 25 Jun 22:33:15.400 # Server started, Redis version 3.2.100
[8032] 25 Jun 22:33:15.400 * The server is now ready to accept connections on port 6379
```

**Run redis-server.exe**

C:\Program Files\Redis\redis-cli.exe

```
127.0.0.1:6379> ping "Hello Majed"
"Hello Majed"
127.0.0.1:6379> ping
PONG
127.0.0.1:6379>
```

ping "Hello Majed"

ping

11:22 PM
11/23/2020

# Reference

- CLI
- Pub/Sub
- Python