



Redis is an open-source key-value NoSQL database that stores its database entirely in the memory, using computer storage to ensure data persistence. As it works primarily from memory (in-memory database), it is extremely fast. It has a client-server architecture (unlike SQLite) with one server capable of serving multiple clients. Redis supports a rich set of data types such as list, set and hashes. Redis can be used for applications that require support for messaging-queues (Publish/Subscribe) and for any short-lived data such as web application sessions, web page hit counts, etc. The number of Redis databases is fixed, and set in the configuration file. By default, you have 16 databases. Each database is identified by a number (not a name) – enter command **INFO**.

## 1. Configuration

- Login through Exeter Virtual Device (WVD)
- Locate S:\ Drive
- Copy the Redis configuration file (**Redis.conf**) and the following two batch files (**redis-cli\_use this.bat** and **redis-server\_use this.bat**) from the [BEMM459J Github repository](#) (under Week 6) to S:\ drive<sup>1</sup>.
- Create folder “Redis” under S:\ drive (this is the folder that will store the database files)
- Start Redis server (**redis-server\_use this.bat**)
- Start Redis client (**redis-cli\_use this.bat**)
- To test server-client connectivity, issue the command “PING” through Redis-client. You will receive the response “PONG” if connectivity had been established (Figure 1). You are now ready to start!
- Redis 127.0.0.1:6379> **INFO**
- Redis 127.0.0.1:6379> **SELECT 1** (database 0-16)
- Redis 127.0.0.1:6379> **DBSIZE** (returns the number of keys in the selected database)

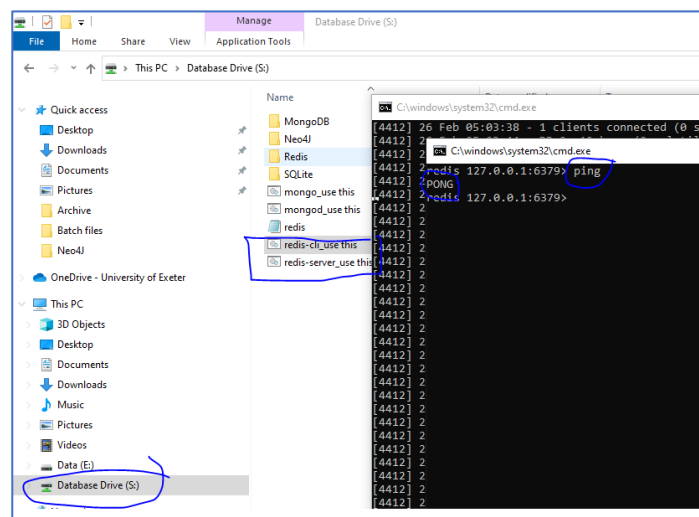


Figure 1: Screenshot of Redis configuration over Exeter WVD.

<sup>1</sup> Note: Redis NoSQL database software is installed in E:\ drive (this is non-persistent). Only S:\ drive (and Windows One Drive) is persistent storage and you must save all files in S:\ drive. Otherwise, you will lose your work between WVD logins.

## 2. Data Types

Redis key commands are used for managing keys. The commands are indicated in **brown font**.

**Redis prompt> COMMAND KEY\_NAME**

### String

A Redis string is a sequence of bytes. Key is *name*. Value is a String *"BEMM459 REDIS Tutorial"*.

**Redis 127.0.0.1:6379> SET** name "BEMM459 REDIS Tutorial"

**Redis 127.0.0.1:6379> GET** name

### Hashes – used to represent objects

A Redis hash is a collection of key-value pairs. Key is *student:1*. Hash data type is used to store the student object, which contains basic information about the student.

**Redis 127.0.0.1:6379> HMSET** student:1 "Hello World" "MSc Business Analytics" "BEMM459" "BEMM458"

**Redis 127.0.0.1:6379> HGETALL** student:1

### Lists

Redis Lists are lists of strings, sorted by insertion order. You can add elements to a Redis List on the head (**LPUSH**) or on the tail (**RPUSH**).

**Redis 127.0.0.1:6379> LPUSH** BEMM459 SQLite (the list is "SQLite")

**Redis 127.0.0.1:6379> LPUSH** BEMM459 Redis (the list is "Redis", "SQLite")

**Redis 127.0.0.1:6379> LPUSH** BEMM459 MongoDB (the list is "MongoDB", "Redis", "SQLite")

**Redis 127.0.0.1:6379> RPUSH** BEMM459 Neo4J (the list is "MongoDB", "Redis", "SQLite", "Neo4J")

**Redis 127.0.0.1:6379> RPUSH** BEMM459 Cassandra (the list is "MongoDB", "Redis", "SQLite", "Neo4J", "Cassandra")

**Redis 127.0.0.1:6379> LRANGE** BEMM459 0 4

### Sets

Redis Sets are an unordered collection of string. A set has "unique" property (no duplicate values allowed).

**Redis 127.0.0.1:6379> SADD** MSc\_BA:BEMM459\_Assessment "Assignment @ 60%"

**Redis 127.0.0.1:6379> SADD** MSc\_BA:BEMM459\_Assessment "Final Exam @ 40%"

**Redis 127.0.0.1:6379> SADD** MSc\_BA:BEMM459\_Assessment "Final Exam @ 40%" (repeating)

**Redis 127.0.0.1:6379> SMEMBERS** MSc\_BA:BEMM459\_Assessment

### Sorted Sets

Redis Sorted Sets are non-repeating collections of Strings. However, every member of a Sorted Set is associated with a score, that is used in order the strings from the smallest to the greatest score. While members are unique, the scores may be repeated.

**Redis 127.0.0.1:6379> ZADD** MSc\_BA:Students 0 "Student A"

**Redis 127.0.0.1:6379> ZADD** MSc\_BA:Students 0 "Student B"

**Redis 127.0.0.1:6379> ZADD** MSc\_BA:Students 5 "Student X"

**Redis 127.0.0.1:6379> ZADD** MSc\_BA:Students 1 "Student Y"

**Redis 127.0.0.1:6379> ZADD** MSc\_BA:Students 1 "Student Y" (repeating)

**Redis 127.0.0.1:6379> ZADD** MSc\_BA:Students 2 "Student Z"

**Redis 127.0.0.1:6379> ZRANGEBYSCORE** MSc\_BA:Students 0 100

## 3. Redis Commands

### GETRANGE

Redis GETRANGE command is used to get the substring of the string value stored at the key, determined by the offsets start and end (both are inclusive).

**Redis 127.0.0.1:6379> SET** name "Hello World"

**Redis 127.0.0.1:6379> GETRANGE** name 0 4

**Redis 127.0.0.1:6379> GETRANGE** name 5 -2

## KEYS

Redis KEYS command is used to search keys with a matching pattern.

**Redis 127.0.0.1:6379> SET** name "Hello"

**Redis 127.0.0.1:6379> SET** name1 "Hello1"

**Redis 127.0.0.1:6379> KEYS** name\* (wild card)

- Note: To get a list of all the keys available in Redis, use only \* (KEYS \*)
- Note: **RANDOMKEY** command is used to get a random key from the database.

## APPEND

Append value to a key.

**Redis 127.0.0.1:6379> SET** name "Hello"

**Redis 127.0.0.1:6379> APPEND** name "World" (returns the length of the string after the append operation)

**Redis 127.0.0.1:6379> GET** name

## EXISTS

Redis EXISTS command is used to check whether the key exists in Redis or not. Returns 1, if the key exists. Returns 0, if the key does not exist.

**Redis 127.0.0.1:6379> EXISTS** Key\_name

## RENAME

Redis RENAME command is used to change the name of a key. It returns an error if the old key and the new key names are equal, or when the key does not exist. If the new key already exists, then it overwrites the existing key.

**Redis 127.0.0.1:6379> RENAME** Key\_name Key\_name\_new

## DEL

The command deletes a key, if it exists.

**Redis 127.0.0.1:6379> DEL** Key\_name

## GETSET

Sets the specified string value in Redis key and returns its old value.

**Redis 127.0.0.1:6379> GETSET** BEMM459:Marks **50** (if the key does not exist then (nil) is returned)

**Redis 127.0.0.1:6379> GETSET** BEMM459:Marks **70** (sets the new value 70 and returns the old value 50)

## SETNX

Sets the specified string value only if key does not exist. **SET** if **Not eXists**.

**Redis 127.0.0.1:6379> SETNX** BEMM459:FinalMarks **50**

**Redis 127.0.0.1:6379> SETNX** BEMM459: FinalMarks **70**

**Redis 127.0.0.1:6379> GET** BEMM459: FinalMarks (returns 50)

- Note: use **MSETNX** to set multiple keys to multiple values, only if none of the keys exist.

## MGET

Gets the value of all specified keys.

**Redis 127.0.0.1:6379> MSET** key1 key2

## SETEX and TTL

Redis SETEX command is used to set some string value with a specified timeout in Redis key (in seconds). Redis TTL command is used to get the remaining time of the key expiry in seconds.

Syntax: **SETEX** KEY\_NAME **TIMEOUT** VALUE

Syntax: **TTL** KEY\_NAME

**Redis 127.0.0.1:6379> SETEX** UOE:StudentGuild:Shop:user34 **60** pen

**Redis 127.0.0.1:6379> TTL** UOE:StudentGuild:Shop:user34 (in seconds; returns -1 when timeout)

**Redis 127.0.0.1:6379> GET** UOE:StudentGuild:Shop:user34 (returns **(nil)** after timeout)

- Note: **PSETEX** command is used to set the value of a key, with the expiration of time in milliseconds (use **PTTL**).
- Note: use **INFO** command to check keys that are set to expire in db0, db1 etc.

### **EXPIRE**

Redis Expire command is used to set the expiry of a key. After the expiry time, the key will not be available in Redis.

**Redis 127.0.0.1:6379> SET name "Hello World"**

**Redis 127.0.0.1:6379> EXPIRE name 45**

**Redis 127.0.0.1:6379> TTL name**

Note: use **PEXPIRE** command to set the expiry in milliseconds (use **PTTL**).

### **EXPIREAT**

Redis Expireat command is used to set the expiry of key in Unix timestamp format. After the expiry time, the key will not be available in Redis.

Syntax: **EXPIREAT KEY\_NAME TIME\_IN\_UNIX\_TIMESTAMP**

Convertor: <https://www.epochconverter.com/>

**Redis 127.0.0.1:6379> SET name "Hello World"**

**Redis 127.0.0.1:6379> EXPIREAT name 1614325238**

**Redis 127.0.0.1:6379> TTL name**

- Note: use **PEXPIREAT** command to set the expiry in milliseconds - **TIME\_IN\_MILLISECONDS\_IN\_UNIX\_TIMESTAMP** (use **PTTL**).

### **PERSIST**

Redis PERSIST command is used to remove the expiration from the key. The return value is either 1 or 0.

Syntax: **PERSIST KEY\_NAME**

**Redis 127.0.0.1:6379> SET name "Hello World"**

**Redis 127.0.0.1:6379> EXPIRE name 120**

**Redis 127.0.0.1:6379> TTL name**

**Redis 127.0.0.1:6379> PERSIST name**

**Redis 127.0.0.1:6379> TTL name** (-1 indicates that the key does not have associated timeout)

### **INCR**

Redis INCR command is used to increment the integer value of a key by one. If the key does not exist, it is set to 0 before performing the operation. An error is returned, if the key contains a value of the wrong type or contains a string that cannot be represented as an integer

**Redis 127.0.0.1:6379> SET BEMM459:Total\_Students 10**

**Redis 127.0.0.1:6379> INCR BEMM459:Total\_Students**

### **INCRBY**

Redis INCRBY command is used to increment the integer value of a key by a specified value. If the key does not exist, it is set to 0 before performing the operation. An error is returned, if the key contains a value of the wrong type or contains a string that cannot be represented as an integer.

**Redis 127.0.0.1:6379> INCRBY BEMM459:Total\_Students 5**

### **DECR**

Redis DECR command is used to decrease the value of a key by one.

**Redis 127.0.0.1:6379> DECR BEMM459:Total\_Students**

### **DECRBY**

Redis DECRBY command is used to decrease the value of a key by a specified value.

**Redis 127.0.0.1:6379> DECR BEMM459:Total\_Students 5**

### MOVE

Redis MOVE command is used to move a key from the currently selected database to the specified destination database.

Return 1 if the key is moved. Returns 0 if the key is not moved.

Syntax: **MOVE** KEY\_NAME DESTINATION\_DATABASE

**Redis 127.0.0.1:6379> SET** mykey "helloworld"

**Redis 127.0.0.1:6379> MOVE** mykey 0 (or other db like 1,4, 12)

- Note: Use INFO command before and after to check move.

### SAVE

Redis BGSAVE command is invoked to manually save data. The "BG" in BGSAVE indicates that the save occurs in the background.

**Redis 127.0.0.1:6379> BGSAVE**

### EXIT

Redis QUIT command is invoked to quit the database.

**Redis 127.0.0.1:6379> QUIT**

References:

- <https://redis.io/commands>
- <https://www.tutorialspoint.com/>

/End