**ACl**

ACL SETUSER john on >hello allcommands -@dangerous +acl|whoami allkeys

Create user john

On means john can login

all command means John have access to all commands

-@dangerous means substract all the dangerous command

+ will allow acl whoami to be excecuted

All keys allow user Diwakar to access all the keys inside the Redis

auth username password // to login with the new user

keys \* is a dangerous command

acl whoami allowed // which user currently logged in

config get \* // not allowed this is an admin command

acl cat dangerous // View all the dangerous commands of REDIS

acl cat // see all categories of commands

acl setuser diwakar on >diwakar +@admin // Setting Diwakar user as admin

acl list // to view all of the users

acl setuser cacheservice on >cache +set +get ~cache:\*

acl deluser Diwakar// it will delete the user

<https://redis.io/docs/management/security/acl/> documentation

**Keys Commands**

· Set Key value

· Get Key

· DEL key1 key2 key3 (To delete a key).

· EXISTS key1 key2 (To check a key exist or not).

· TTL key  (To check time to live).

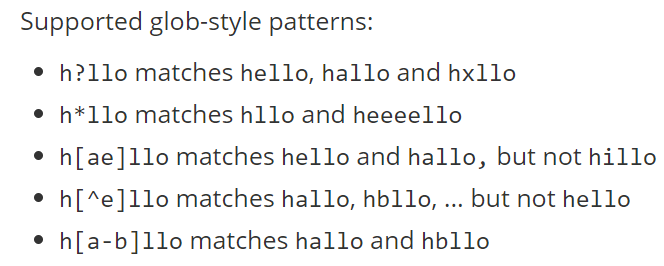
· EXPIRE key 10(in seconds).

· PTTL mykey (to check time in millisecond).

· PEXPIRE mykey 1500 (Time in Milliseconds).

· PERSIST mykey (Remove EXPIRATION from the key)

· KEYS a?? (Returns all keys matching pattern)



· RANDOMKEY (Return a random key from the currently selected database)

· RENAME mykey myotherkey

· RENAMENX mykey myotherkey (Renames key to newkey if newkey does not yet exist)

· TOUCH key1 key2 (Alters the last access time of a key(s).

· UNLINK key1 key2 key3 (The actual removal will happen later asynchronously.)

· TYPE key1 (Return Type of Value)

· DUMP mykey (Serialize the value stored at the key in a Redis-specific format)

· RESTORE mykey 0 "\n\x17\x17\x00\x00\x00\x12\x00\x00\x00\x03\x00\”

**Strings**

 APPEND mykey "Hello" (appends the value at the end of the string).

· INCR mykey (Increments the number stored at key by one).

· INCRBY mykey 5

· DECR mykey (Decrements the number stored at key by one.)

· DECRBY mykey 3 (Decrements the number stored at key )

· INCRBYFLOAT mykey 0.1(Increment the string representing a floating point number)

· GETSET mycounter "0" (Atomically sets key to value and returns the old value stored at key)

· MSET key1 "Hello" key2 "World" (Sets the given keys to their respective values.)

· MGET key1 key2 (Returns the values of all specified keys).

· SETNX mykey "Hello" (Set key to hold string value if key does not exist.)

· MSETNX key1 "Hello" key2 "there" (Sets the given keys to their respective values.)

· GETRANGE mykey 0 3 (Returns the substring of the string value stored at key).

· GETRANGE mykey -3 -1. (-1 means the last character and so on).

· SETEX mykey 10 "Hello" (Set key to hold the string value and set key to timeout after a given number of seconds.)

· PSETEX mykey 1000 "Hello" ( expire time is specified in milliseconds instead of seconds.)

· SETRANGE key1 6 "Redis" (Overwrites part of the string stored at key) .

· STRLEN mykey ( Returns the length of the string )

**Lists**

**RPUSH mylist "hello"** (Insert all the specified values at the tail of the list stored at key.)

**LRANGE mylist 0 -1** (Returns the specified elements of the list stored at key.) -1 means end of the list

**LPUSH mylist "world"** (Insert all the specified values at the head of the list stored at key.)

**RPUSHX mylist "World"** (Inserts value at the tail of the list stored at key, only if key already exists and holds a list.)

**LPUSHX mylist "World"**(Inserts value at the head of the list stored at key, only if key already exists and holds a list.)

**RPOP mylist** (Removes and returns the last element of the list stored at key.) take the element process it becaue you don’t care after that

**LPOP mylist** (Removes and returns the first element of the list stored at key.)

**LTRIM mylist 0 2 first three elements would be left** (Trim an existing list so that it will contain only the specified range of elements specified) LTRIM foobar 0 2 will modify the list stored at foobar so that only the first three elements of the list will remain

**LSET mylist 0 "four"** (Sets the list element at index to value.) update the val in the list

**LINDEX mylist 0** (Returns the element at index in the list stored at key.)

**LINSERT mylist BEFORE "World" "There"** (Inserts value in the list stored at key either before or after the reference value pivot.)

**LLEN mylist** (Returns the length of the list stored at key. )

**LREM mylist 2 "hello"** (Removes the first count occurrences of elements equal to value from the list stored at key.)

**Hash**

**HSET myhash field1 "Hello"**.  (Sets field in the hash stored at key to value. If field already exists in the hash, it is overwritten.)

**HGET myhash field1** (Returns the value associated with field in the hash stored at key.)

**HGETALL myhash**  (Returns all fields and values of the hash stored at key.)

**HMGET myhash field1 field2**(Returns the values associated with the specified fields in the hash stored at key.)

**HEXISTS myhash field1** (Returns if field is an existing field in the hash stored at key.)

**HKEYS myhash** (Returns all field names in the hash stored at key.)

**HLEN myhash** (Returns the number of fields contained in the hash stored at key.)

**HSETNX myhash field "Hello"** (Sets field in the hash stored at key to value, only if field does not yet exist.)

**HDEL myhash field1** (Removes the specified fields from the hash stored at key.)

**HINCRBY myhash field 1** (Increments the number stored at field in the hash stored at key by increment.)

**HINCRBYFLOAT mykey field 0.1** (Increment the specified field of a hash stored at key, and representing a floating point number, by the specified increment.)

**HSTRLEN myhash f1** (Returns the string length of the value associated with field in the hash stored at key.)

**HVALS myhash** (Returns all values in the hash stored at key.)

**Sets**

**SADD myset "Hello"** Add the specified members to the set stored at key.

**SMEMBERS myset** (Returns all the members of the set value stored at key.)

**SISMEMBER myset "one”** (Returns if member is a member of the set stored at key.)

**SCARD myset**(Returns the set cardinality (number of elements) of the set stored at key.)

**SPOP myset** (Removes and returns one or more random elements from the set value store at key.)

**SREM myset "one"** (Remove the specified members from the set stored at key.)

**Sorted Set**

**ZADD myzset 2 "two" 3 "three"** (<https://redis.io/commands/zadd>  Refer this link to know more.)

**ZRANGE myzset 0 -1 WITHSCORES** (Returns the specified range of elements in the sorted set stored at key.)

**ZCARD myzset**(Returns the sorted set cardinality (number of elements) of the sorted set stored at key.)

**ZREM myzset "two"** (Removes the specified members from the sorted set stored at key. Non existing members are ignored.)

**ZSCORE myzset "one"**(Returns the score of member in the sorted set at key.)

**ZINCRBY myzset 2 "one"** (Increments the score of member in the sorted set stored at key by increment.) if element doesn’ exist it will create that element

**ZREVRANGE myzset 0 -1**(Returns the specified range of elements in the sorted set stored at key. The elements are considered to be ordered from the highest to the lowest score.)

**ZRANK myzset "three"**(Returns the rank of member in the sorted set stored at key, with the scores ordered from low to high. The rank (or index) is 0-based, which means that the member with the lowest score has rank 0.) to check who is the last

**ZREVRANK myzset "one"** (Returns the rank of member in the sorted set stored at key, with the scores ordered from high to low. The rank (or index) is 0-based, which means that the member with the highest score has rank 0.)

**ZCOUNT myzset -inf +inf** (Returns the number of elements in the sorted set at key with a score between min and max.)

**ZRANGEBYSCORE myzset (1 2**(Returns all the elements in the sorted set at key with a score between min and max)

**zrange players:exp 1 4 Byscore withscores** fetch the players on the basis if the score falls in between the range

**ZREVRANGEBYSCORE myzset 2 (1**  (Apart from the reversed ordering, ZREVRANGEBYSCORE is similar to ZRANGEBYSCORE.)

**ZPOPMAX myzset**(Removes and returns up to count members with the highest scores in the sorted set stored at key.)

**ZPOPMIN myzset** (Removes and returns up to count members with the lowest scores in the sorted set stored at key.

**ZREMRANGEBYRANK myzset 0 1** (Removes all elements in the sorted set stored at key with rank between start and stop.)

**ZREMRANGEBYSCORE myzset -inf (2**  (Removes all elements in the sorted set stored at key with a score between min and max (inclusive).)

**Streams**

**xadd temprature NOMKSTREAM \* tempF 43.7 //** creates a stream and make sure the stream exist with NOMKSTREAM option

**xrange temprature -+ //** fetch all stream data

**xadd manual\_id\_stream 101-1 foo bar. //** Just ensure your ids are always increasing

**xlen num** // how many ids are stored in stream

**xtrim numbers MINID 1639994900038-0 //** any older msg id would be removed , based on the time

**xtrim man maxlen 100. //** keep latest 100 elements

**xrevrange numbers + - count 5 //** latest to oldest

**xrevrange numbers 1639995941975-0 1639995941974-**9 // newest : oldest

**xread count 3 streams numbers 0-0 //** give 3 elements after 0-0 id

**xread count 1 block 1000 streams numbers 1639997101522-7 //** block the consumer for 1 second and wait for the new msg after the specified msg id.

**xread count 1 block 0 streams numbers $ //** $ is used to refer last msg id. Use $ once in your api call after that use the returned id for future data fetching