Redis-cli

Redis-cli (Redis command line interface)

Redis-cli 사용법을 설명합니다.
이 문서는 버전 4.0.9를 기준으로 작성했습니다.

명령 목록

- USER User와 password를 사용합니다.
- PIPE 대량 데이터를 읽어들일(load) 때 사용합니다.
- RDB RDB 파일을 다른 파일 이름으로 저장합니다.
- STAT 레디스 서버의 중요 통계정보를 주기적으로 보여줍니다.
- MON 모니터용 레디스 서버의 중요 통계정보를 주기적으로 보여줍니다.
- BIGKEYS 데이터 타입별로 큰 키를 찾아 보여줍니다.
- SCAN 키를 scan 해서 보여줍니다.
- SLAVE Redis-cli의 역할을 복제서버(슬레이브)로 변경하여 마스터로 부터 어떤 명령들이 오는지 관찰할 수 있습니다.
- CSV 명령 실행 결과를 CSV(Comma Separated Values) 형태로 출력합니다.
- Command 명령을 실행합니다.
- 한글 사용
- Interactive mode 대화식으로 명령을 실행합니다.
- LATENCY 응답시간을 연속해서 보여준다.
- LATENCY-HISTORY 응답시간을 연속해서 보여주는데 15초가 지나면 새 줄에 보여준다.
- LATENCY-DIST 응답시간을 색이 있는 그래프로 보여준다.
- INTRINSIC-LATENCY 연산을 수행해서 얻은 응답시간을 기준으로 보여준다.
- HELP 도움말을 보여준다.

USER

- 버전 6.0부터 ACL이 적용되어 user와 password를 입력할 수 있습니다. 첫 번째와 두 번째처럼 사용할 수 있으나, 이 경우 command line에서 입력하므로 패스워드가 로그로 남아 유출될 수 있다. 그래서 세 번째처럼 사용하면 로그로 인한 유출 위험이 없어진다.

```
$ src/redis-cli -p 6000 --user user1 --pass password
Warning: Using a password on the command line interface may not be safe.
127.0.0.1:6000>

$ src/redis-cli -p 6000 --user user1 -a password
Warning: Using a password on the command line interface may not be safe.
127.0.0.1:6000>
```

```
$ src/redis-cli -p 6000 --user user1 --askpass
Please input password: ********
127.0.0.1:6000>
```

PIPE

- AOF 파일 또는 명령이 있는 txt 파일을 읽어들일(load) 때 사용합니다. Redis.conf에 지정된 AOF 파일은 레디스 서버가 시작할 때 읽어들입니다. Pipe는 다른 서버의 AOF 파일을 로드할 때 사용하거나 레디스 명령이 있는 txt 파일로 부터 데이터를 받을 때 사용합니다.
- 소문자(pipe)를 사용합니다.
- AOF 파일 저장은 BGREWRITEAOF 명령을 사용합니다.
- 사용법: cat appendonly.aof | redis-cli -h ip -p port --pipe

```
$ cat data.aof | src/redis-cli -p 6001 --pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 1001
```

- 시험용 명령 파일(1백만개 명령) 만들고 pipe로 로드하기

```
$ for i in {000000..999999}; do echo set key$i value$i >> data.txt; done
$ cat data.txt | redis-cli -p 6000 --pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 1000000
```

RDB

- RDB 파일을 다른 파일 이름으로 저장합니다.
- Redis.conf에 지정된 파일명으로 저장하는 것은 SAVE, BGSAVE 명령을 사용합니다.
- 내부적으로 Sync 방식인 SAVE 명령을 사용하므로 바쁜 서버에서는 사용에 주의해야 합니다.
- 사용법: redis-cli --rdb dump.rdb

```
$ src/redis-cli -p 6001 --rdb dump.rdb
SYNC sent to master, writing 35753 bytes to 'dump.rdb'
Transfer finished with success.
```

- 스크립트나 cron job에서 사용할 때는 **echo $?**로 결과를 확인하는 것이 좋습니다. 정상이면 '0'이고 에러가 발생하면 '0' 이외의 값이 나옵니다.

STAT

- 레디스 서버의 중요 통계정보를 주기적으로 보여준다.
- 사용법: $ src/redis-cli -p 6001 --stat

```
$ src/redis-cli -p 6001 --stat
------- data ------ -------------------- load -------------------- - child -
keys     mem clients blocked requests connections
2002   1.69M   1       0       8241 (+1)           7113
2177   1.72M   1       0       8417 (+176)         7288
2662   1.75M   1       0       8903 (+486)         7773
3140   1.78M   1       0       9382 (+479)         8251
3620   1.87M   2       0       9863 (+481)         8731
4130   1.88M   1       0      10374 (+511)         9241
4608   1.91M   1       0      10853 (+479)         9719
5084   2.00M   2       0      11330 (+477)        10195
5611   1.98M   1       0      11858 (+528)        10722
```

- **-i interval(sec)** 옵션으로 주기를 설정할 수 있다. 기본은 1초이다.

- **--pagesize 30** 옵션으로 page size를 설정할 수 있다. 기본은 20이다.
  Page Header를 맨 위에 한번만 찍게 하려면 0으로 설정하세요.
- **-t** 옵션으로 일시(date time)를 찍을 수 있습니다.

  $ src/redis-cli -p 6001 **-t** --stat
  ---------- date ---------- ------- data ------ ---------- load ---------- - child -
  date time keys mem clients blocked requests connections
  2019-08-31 10:58:39 2002 1.69M 1 0 8241 (+1) 7113
  2019-08-31 10:58:39 2177 1.72M 1 0 8417 (+176) 7288
  2019-08-31 10:58:39 2662 1.75M 1 0 8903 (+486) 7773

- Pagesize와 t(일시) 옵션은 **Redis Enterprise 버전**에서 사용 가능합니다.
- 내부적으로 INFO 명령을 사용하고, 각 항목의 정보는 다음과 같다.
  - **keys**: 총 키 개수, KEYSPACE의 키(keys) 합계이다.
  - **mem**: used_memory MEMORY section - 현재 사용 메모리
  - **clients**: connected_clients CLIENTS section - 연결된 클라이언트 수
  - **blocked**: blocked_clients CLIENTS section - BLPOP 같은 명령으로 대기중인 클라이언트 수
  - **requests**: total_commands_processed STATS section - 처리한 총 명령 수
  - **connections**: total_connections_received STATS section - 총 접속 수
  - **child**: 자식 프로세스가 실행되어서 AOF나 RDB 파일을 저장중이거나, 로딩 중이면 표시된다.
    - BGSAVE(bgsave_in_progress)가 실행중이면 SAVE
    - BGREWRITEAOF(aof_rewrite_in_progress)가 실행중이면 AOF
    - AOF/RDB(loading) 파일을 로딩중이면 LOAD

STAT(통계)를 파일로 남기는 방법

- **redis-cli -p 6001 -i 60 -t --stat > redis_stat.log &**
  이렇게 하면 백그라운드로 실행되고 60초마다 일시를 포함해서 redis_stat.log 파일에 서버 정보를 남깁니다.
- 그런데 한가지 문제가 있습니다. 리눅스는 이런 경우 4K바이트 한 메모리 페이지가 꽉차야 디스크에 씁니다. 그래서 파일에 기록된 것을 보려면 한참 기다려야 합니다. 이것을 해결하려면 다음과 같이 사용합니다.
- stdbuf -oL **redis-cli -p 6001 -i 60 -t --pagesize 0 --stat > redis_stat.log &**
  Stdbuf -oL을 사용하면 바로 디스크에 씁니다. **oL** 옵션은 output을 Line 단위로 처리하겠다는 것입니다. 그래서 printf로 한 라인 프린트할 때마다 디스크에 쓰여(flush)집니다.
  Stdbuf를 사용하면 tail -f redis_stat.log 같은 명령으로 바로 확인할 수 있습니다.

MON Monitor

- 모니터용 레디스 서버의 중요 통계정보를 주기적으로 보여준다.
- 이 기능은 **Enterprise 버전**에서 사용 가능합니다.
- 사용법: $ src/redis-cli -p 6001 --mon

  ---
  date time ip:port role keys memory clients commands from to replicas child
  ---
  2020-07-03 13:37:09 18:30010 master 5272026 1.55G 1 1 0 0 0 LOAD
  2020-07-03 13:37:10 18:30010 master 5464945 1.60G 1 1 0 0 0 LOAD
  2020-07-03 13:37:11 18:30010 master 5657875 1.62G 1 1 0 0 0 LOAD
  2020-07-03 13:37:12 18:30010 master 5872193 1.68G 1 1 0 0 0 LOAD
  2020-07-03 13:37:13 18:30010 master 6086435 1.74G 1 1 0 0 0 LOAD

- 이것은 info mon 명령을 실행해서 얻은 값을 보여주는 것이다. 위에서 설명한 stat는 info 명령을 실행하는데 info mon이 그냥 info 보다 5배 이상 빠르게 실행된다.
- ip:port에서 ip는 ip의 네 번째 숫자를 보여준다.
- role은 master 또는 replica이다.

- from과 to는 Enterprise 버전에서 Active-Active로 운영할 경우 데이터를 받는 서버 수, 보내는 서버 수이다.
- replicas는 master-replica 관계에서 복제노드 수이다.
- child는 ① rdb 또는 AOF 파일을 로드할 경우(레디스 서버 시작 시) **LOAD**,
  ② rdb 파일을 저장할 경우 **SAVE**,
  ③ AOF 파일을 저장(rewrite)할 경우 **AOF**로 표시된다.

BIGKEYS

- 데이터 타입별로 큰 키를 찾아 보여줍니다. 키에 대한 통계 정보를 볼 수 있습니다.
- 내부적으로 SCAN 명령을 사용하므로 서버에 큰 영향을 주지 않습니다.
- 아래 예에서 summary 부터 설명하면, String은 'keyFFFF'의 value가 364 바이트로 가장 크고, List는 'mylistB'가 100 items으로 가장 크고, Set은 'mysetB'가 300 members로 가장 큰것을 보여줍니다. 다음은 각 데이터 타입별로 키 개수와 bytes/items/members/fields 수를 보여줍니다.

```
$ src/redis-cli -p 6001 --bigkeys

# Scanning the entire keyspace to find biggest keys as well as
# average sizes per key type. You can use -i 0.1 to sleep 0.1 sec
# per 100 SCAN commands (not usually needed).

[00.00%] Biggest string found so far 'keyB2810' with 10 bytes
[10.53%] Biggest list found so far 'mylistB' with 100 items
[19.10%] Biggest string found so far 'keyFFFF' with 364 bytes
[44.96%] Biggest set found so far 'myset' with 100 members
[91.42%] Biggest set found so far 'mysetB' with 200 members

-------- summary -------

Sampled 7007 keys in the keyspace!
Total key length in bytes is 51718 (avg len 7.38)

Biggest string found 'keyFFFF' has 364 bytes
Biggest list found 'mylistB' has 100 items
Biggest set found 'mysetB' has 200 members

7003 strings with 66063 bytes (99.94% of keys, avg size 9.43)
2 lists with 110 items (00.03% of keys, avg size 55.00)
2 sets with 300 members (00.03% of keys, avg size 150.00)
0 hashs with 0 fields (00.00% of keys, avg size 0.00)
0 zsets with 0 members (00.00% of keys, avg size 0.00)
0 streams with 0 entries (00.00% of keys, avg size 0.00)
```

SCAN

- 키를 scan해서 보여줍니다. --pattern 옵션으로 패턴을 사용할 수 있습니다.
- Head, more, wc 같은 리눅스 명령과 조합해서 다양하게 사용할 수 있습니다.

```
$ src/redis-cli -p 6001 --scan --pattern 'keyB*' | wc
5001 5001 43898
```

SLAVE

- Redis-cli의 역할을 복제서버(슬레이브)로 변경하여 마스터로 부터 어떤 명령들이 오는지 관찰할 수 있습니다.
- 지정한 서버의 복제서버처럼 동작한다. 주로 디버그 용도로 사용합니다.

```
$ src/redis-cli -p 6001 --slave
SYNC with master, discarding 141585 bytes of bulk transfer...
SYNC done. Logging commands from master.
```

```
"PING"
"SELECT","0"
"set","key","value"
"PING"
```

CSV

- 명령 실행 결과를 CSV(Comma Separated Values) 형태로 출력합니다.

```
$ src/redis-cli -p 6001 --csv smembers myset
"Value37","Value17","Value44", ..., "Value87"
```

Command

- 레디스 명령을 바로 실행합니다.
- --raw 옵션을 사용하면 "(integer)"가 나오지 않습니다.
- '>' 파일 출력을 사용하면 --raw 옵션을 사용하지 않아도 숫자만 저장됩니다.
- '>' 파일 출력에 "(integer)"도 저장하고 싶으면 --no-raw 옵션을 사용합니다.

```
$ src/redis-cli -p 6001 incr mycountor
(integer) 1
$ src/redis-cli -p 6001 --raw incr mycountor
2
$ src/redis-cli -p 6001 incr mycountor > my.txt
$ cat my.txt
3
$ src/redis-cli -p 6001 --no-raw incr mycountor > my.txt
$ cat my.txt
(integer) 4
```

- 명령을 반복 실행하고 싶으면 **-r 숫자** 옵션을 사용합니다.

```
$ src/redis-cli -p 6001 -r 5 incr mycountor
(integer) 5
(integer) 6
(integer) 7
(integer) 8
(integer) 9
$ src/redis-cli -p 6001 -r 5 lpush mylist AAAAA
(integer) 11
(integer) 12
(integer) 13
(integer) 14
(integer) 15
```

- **-i 숫자(sec)** 옵션을 사용하면 interval를 줄 수 있습니다.

```
$ src/redis-cli -p 6001 -r 5 -i 3 incr mycountor
```

- **-r -1**를 사용하면 계속 반복할 수 있다. 이 기능은 버전 4.0.9까지 사용가능합니다. 무한 반복하면 overflow가 발생합니다. 버전 4.0.10부터는 사용할 수 없도록 수정되었습니다.

```
$ src/redis-cli -p 6001 -r -1 -i 1 info| grep rss_human
used_memory_rss_human:7.80M
used_memory_rss_human:7.80M
^C
```

- 리눅스 쉘 기능을 사용하면 변수(숫자)를 사용할 수 있다. 이렇게 하면 key와 value에 숫자를 붙여서 바꾸면서 SET 명령을 실행할 수 있다. 이것은 redis-cli 자체가 여러 번 실행되는 것입니다.

```
$ for i in {1..10}; do src/redis-cli -p 6001 set key$i value$i; done
OK
```

...
OK

- 접속 port를 변수로 사용할 수도 있습니다. 여러 서버에 명령을 일괄 실행할 때 유용합니다.

```
$ for i in {6001..6010}; do src/redis-cli -p $i dbsize; done
(integer) 7011
(integer) 4231
...
(integer) 8654
$ for i in {6001..6010}; do src/redis-cli -p $i config rewrite; done
OK
...
OK
```

한글 사용

- **--raw** 옵션을 사용하면 한글을 사용할 수 있습니다.

```
$ src/redis-cli -p 6001 --raw
127.0.0.1:6001> set key "한글"
OK
127.0.0.1:6001> get key
한글
```

- Redis Enterprise에서 제공하는 redis-ecli의 경우 raw 옵션없이도 한글을 사용할 수 있습니다.

Interactive mode

- **help <commandname>**

```
127.0.0.1:6001> help zadd
  ZADD key [NX|XX] [CH] [INCR] score member [score member ...]
  summary: Add one or more members to a sorted set, or update its score if it already
exists
  since: 1.2.0
  group: sorted_set
```

- **help @<category>** Category를 입력하면 데이터 타입별 명령 리스트를 볼 수 있습니다. Category는 generic, list, set, sorted_set, hash, pubsub, transactions, connection, server, scripting, hyperloglog 가 있습니다.

```
127.0.0.1:6001> help @connection
 AUTH password
 summary: Authenticate to the server
 since: 1.0.0

 ECHO message
 summary: Echo the given string
 since: 1.0.0

 PING [message]
 summary: Ping the server
 since: 1.0.0

 QUIT -
 summary: Close the connection
 since: 1.0.0

 SELECT index
 summary: Change the selected database for the current connection
 since: 1.0.0
```

- 반복 실행할 수 있습니다.

```
127.0.0.1:6001> 3 incr mycountor
(integer) 10
(integer) 11
(integer) 12
```

- **Clear** 화면을 지웁니다.
- 명령 앞 글자를 입력하고 **<TAB>**을 치면 명령이 완성됩니다.

```
127.0.0.1:6001> Z<TAB> -> ZADD
```

- **화살표**를 사용하면 이전/이후 명령을 볼 수 있습니다. 명령은 '.rediscli_history' 파일에 저장됩니다.

LATENCY

- 10ms마다(1초에 100번) PING을 보내서 얻은 응답시간이다. 응답시간 최소(min), 최대(max), 평균(avg), PING 보낸 횟수를 보여준다. 시간 단위는 millisecond이고 samples가 PING 횟수이다. 멈추려면 ctrl+C를 누른다.
- static void latencyMode(void) 함수가 실행된다.

```
$ redis-cli -p 6001 --latency
min: 0, max: 1, avg: 0.14 (15863 samples)
```

LATENCY-HISTORY

- 보여주는 내용은 latency와 같고, 주어진 시간 단위로 새 줄에 보여준다. 디폴트는 15초이고 -i second 옵션을 사용해서 간격을 조정할 수 있다. 시간 단위는 millisecond이다.

```
# redis-cli -p 7124 --latency-history -i 5
min: 0, max: 1, avg: 0.14 (488 samples) -- 5.00 seconds range
min: 0, max: 1, avg: 0.14 (488 samples) -- 5.01 seconds range
min: 0, max: 1, avg: 0.13 (488 samples) -- 5.00 seconds range
min: 0, max: 1, avg: 0.16 (187 samples)^C
```

LATENCY-DIST

- 바탕색이 비율(%)이고 문자가 시간을 나타낸다. 시간단위 microsecond이다. PING 응답시간이 기준이다. dist는 distribution의 약자이다.
- static void latencyDistMode(void) 함수가 실행된다.

```
# redis-cli -p 7124 --latency-dist : 1초 간격으로 보여준다.
# redis-cli -p 7124 --latency-dist -i 5 : 5초 간격으로 보여준다.
```

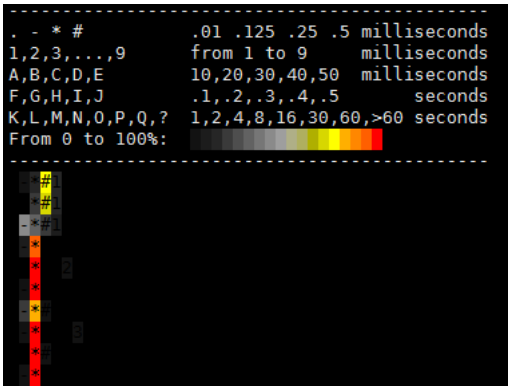

그림 2-1 Latency distribution mode graph

- 문자는 PING 응답시간이 0.01ms 이하일 때 '.', 0.125ms 이하일 때 '-' 등으로 표시된다.
  바탕색은 응답시간의 비율을 나타낸다. 칸(색)이 19개 이므로 약 5% 단위이다.
  첫 번째 줄을 보면 샾(#)은 0.25ms < 응답시간 <= 0.5ms가 약 75%를 차지하고, 별(*)은 0.125ms < 응답시간 <= 0.25ms가 10%, 1은 0.5ms < 응답시간 <= 1ms가 10%, 거의 보이지 않지만 바(-)는 0.01ms < 응답시간 <= 0.125ms가 5%를 뜻한다.

INTRINSIC-LATENCY

- 이것은 redis-cli에서 연산을 수행해서 얻은 응답시간을 기준으로 보여준다. CPU 성능을 체크하는 것이라고 보는 편이 적절하겠다. redis-cli 수행 시 서버를 지정할 필요가 없다. redis-cli에서 수행되는 것이므로 서버와 같은 박스(머신/VM)에서 수행해야 한다.
- static void intrinsicLatencyMode(void) 함수가 실행된다.
  아래는 10초 동안 수행한 결과이다.

```
# redis-cli --intrinsic-latency 10
Max latency so far: 1 microseconds.
Max latency so far: 3 microseconds.
Max latency so far: 6 microseconds.
Max latency so far: 157 microseconds.

234859166 total runs (avg latency: 0.0426 microseconds / 42.58 nanoseconds per run).
Worst run took 3687x longer than the average latency.
```

- 10초 동안 234,859,166회 수행되었고, 수행 1회당 42.58 nanoseconds가 걸렸다.
  아래는 실제 수행되는 코드이다.

```
unsigned long compute_something_fast(void) {
    unsigned char s[256], i, j, t;
    int count = 1000, k;
    unsigned long output = 0;

    for (k = 0; k < 256; k++) s[k] = k;

    i = 0;
    j = 0;
    while(count--) {
        i++;
        j = j + s[i];
        t = s[i];
        s[i] = s[j];
        s[j] = t;
        output += s[(s[i]+s[j])&255];
    }
    return output;
}
```

기타

- DB를 지정해서 접속하려면 -n 옵션을 사용합니다. 그러면 지정한 DB에 바로 접속할 수 있습니다.

```
$ src/redis-cli -p 6000 -n 3
127.0.0.1:6000[3]>
```

이 기능은 shell에서 redis-cli에 바로 접속해서 명령을 실행할 경우 유용합니다.
3번 DB에 데이터를 넣으려고 이렇게 2개의 명령을 연속해서 실행할 수 없습니다.

```
$ src/redis-cli select 3, set key value
```

또한 이렇게 실행하면 두 번째 명령은 0번 DB에서 실행됩니다.

```
$ src/redis-cli select 3
$ src/redis-cli set key value
```

HELP

```
$ src/redis-cli --help
redis-cli 4.0.9

Usage: redis-cli [OPTIONS] [cmd [arg [arg ...]]]
  -h <hostname>      Server hostname (default: 127.0.0.1).
  -p <port>          Server port (default: 6379).
  -s <socket>        Server socket (overrides hostname and port).
  -a <password>      Password to use when connecting to the server.
  -u <uri>           Server URI.
  -r <repeat>        Execute specified command N times.
  -i <interval>      When -r is used, waits <interval> seconds per command.
                     It is possible to specify sub-second times like -i 0.1.
  -n <db>            Database number.
  -x                 Read last argument from STDIN.
  -d <delimiter>     Multi-bulk delimiter in for raw formatting (default: \n).
  -c                 Enable cluster mode (follow -ASK and -MOVED redirections).
  --raw              Use raw formatting for replies (default when STDOUT is
                     not a tty).
  --no-raw           Force formatted output even when STDOUT is not a tty.
  --csv              Output in CSV format.
  --stat             Print rolling stats about server: mem, clients, ...
  --latency          Enter a special mode continuously sampling latency.
                     If you use this mode in an interactive session it runs
                     forever displaying real-time stats. Otherwise if --raw or
                     --csv is specified, or if you redirect the output to a non
                     TTY, it samples the latency for 1 second (you can use
                     -i to change the interval), then produces a single output
                     and exits.
  --latency-history  Like --latency but tracking latency changes over time.
                     Default time interval is 15 sec. Change it using -i.
  --latency-dist     Shows latency as a spectrum, requires xterm 256 colors.
                     Default time interval is 1 sec. Change it using -i.
  --lru-test <keys>  Simulate a cache workload with an 80-20 distribution.
  --slave            Simulate a slave showing commands received from the master.
  --rdb <filename>   Transfer an RDB dump from remote server to local file.
  --pipe             Transfer raw Redis protocol from stdin to server.
  --pipe-timeout <n> In --pipe mode, abort with error if after sending all data.
                     no reply is received within <n> seconds.
                     Default timeout: 30. Use 0 to wait forever.
  --bigkeys          Sample Redis keys looking for big keys.
  --hotkeys          Sample Redis keys looking for hot keys.
                     only works when maxmemory-policy is *lfu.
  --scan             List all keys using the SCAN command.
  --pattern <pat>    Useful with --scan to specify a SCAN pattern.
  --intrinsic-latency <sec> Run a test to measure intrinsic system latency.
                     The test will run for the specified amount of seconds.
  --eval <file>      Send an EVAL command using the Lua script at <file>.
  --ldb              Used with --eval enable the Redis Lua debugger.
  --ldb-sync-mode    Like --ldb but uses the synchronous Lua debugger, in
                     this mode the server is blocked and script changes are
                     are not rolled back from the server memory.
  --help             Output this help and exit.
  --version          Output version and exit.

Examples:
  cat /etc/passwd | redis-cli -x set mypasswd
  redis-cli get mypasswd
  redis-cli -r 100 lpush mylist x
  redis-cli -r 100 -i 1 info | grep used_memory_human:
  redis-cli --eval myscript.lua key1 key2 , arg1 arg2 arg3
  redis-cli --scan --pattern '*:12345*'

  (Note: when using --eval the comma separates KEYS[] from ARGV[] items)

When no command is given, redis-cli starts in interactive mode.
```

```
Type "help" in interactive mode for information on available commands
and settings.
```

<< Redis Server Performance Redis-cli Redis-benchmark >>

redisgate@gmail.com 02.503.2235
서울시 강남구 강남대로 342 역삼빌딩 5층 (역삼동) 우 06242