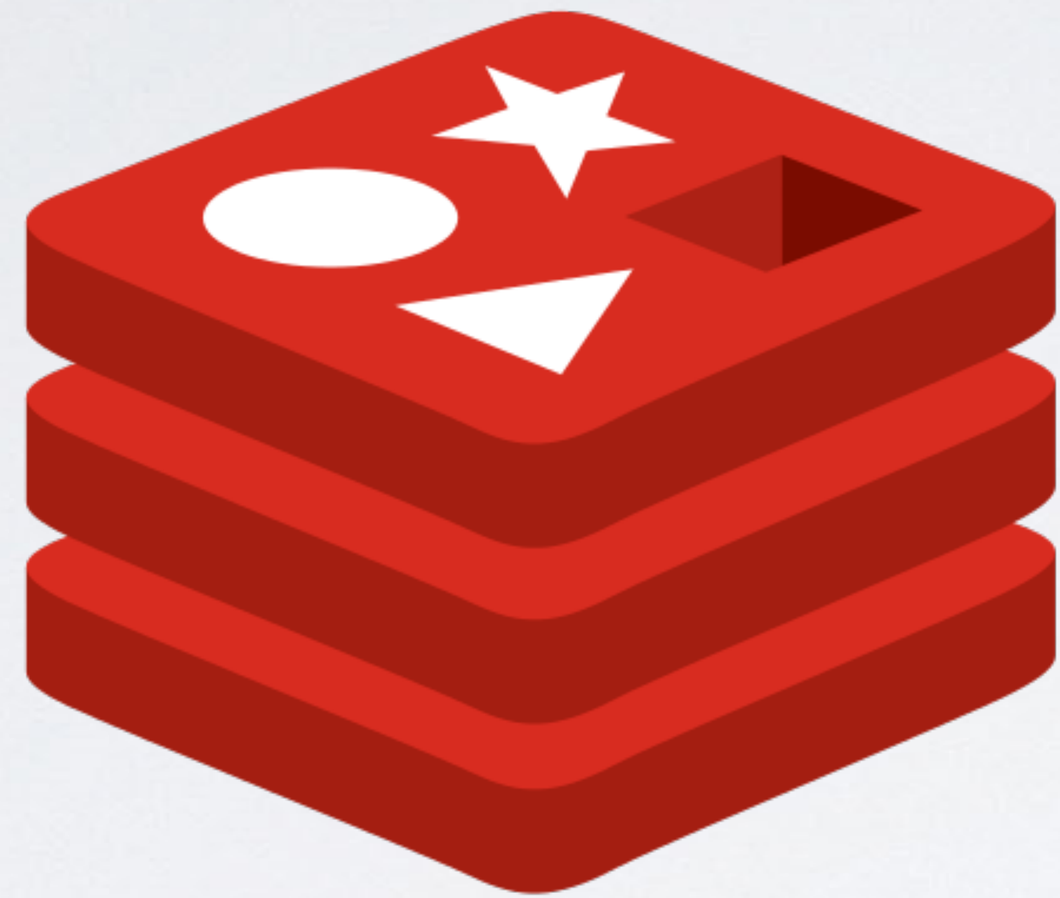


redis
Fundamental

acoshift (2017)

Redis

REmote DIctionary Server

REDIS ?

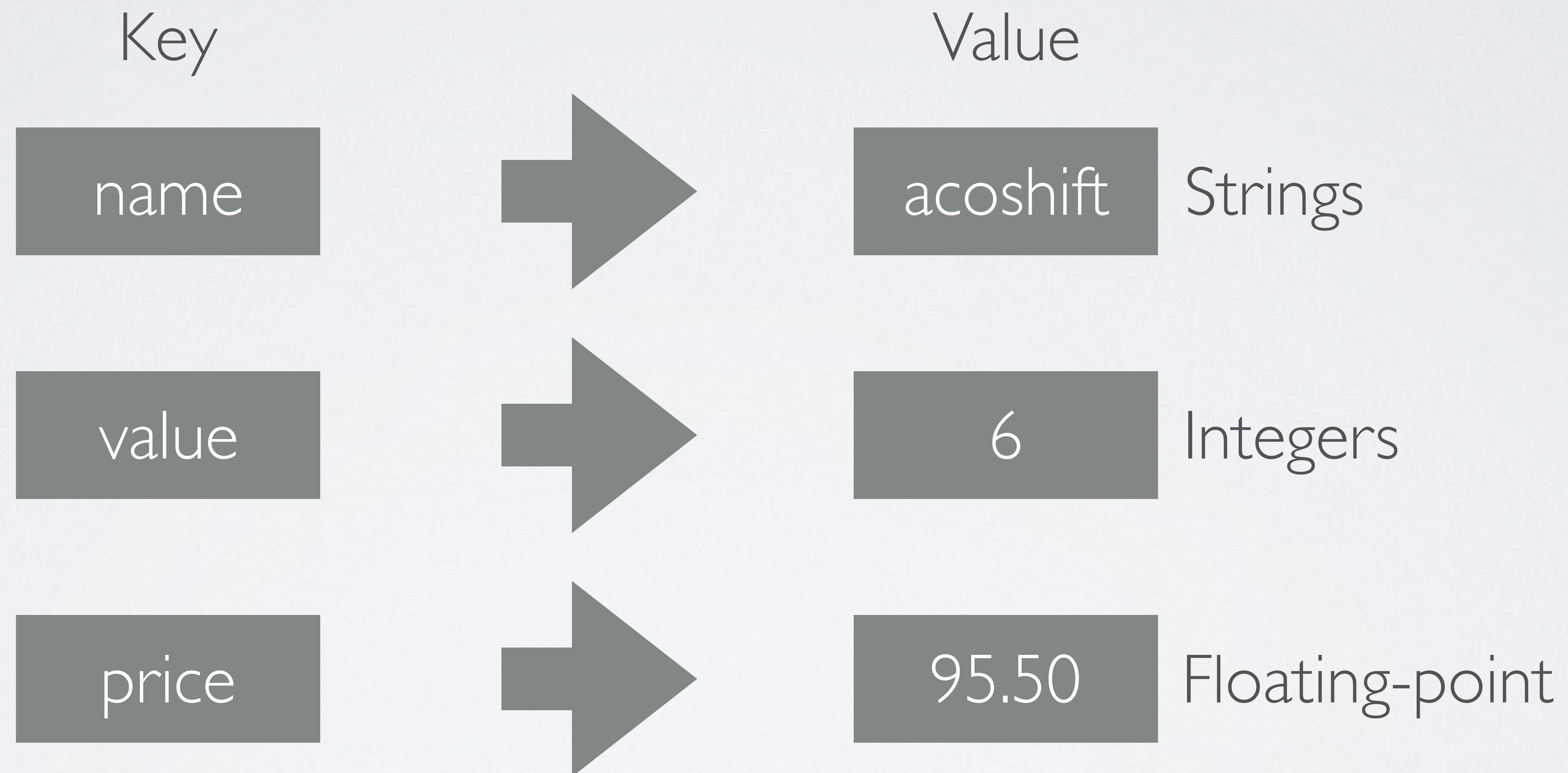
- Key-Value data structure store
- Configurable persistent
- Support atomic operations
- Support transactions
- Single threaded

DATA TYPES

- Strings
- Lists
- Sets
- Hashes
- Sorted Set
- Bitmap
- HyperLogLogs
- GEO
- Pub/Sub

STRING

(Strings, Integers, Floating-point)



HASH TABLE

h("name") = 2

h("value") = 0

h("price") = 3

T

0	6
1	
2	acoshift
3	95.50
4	

$h(k) = k \bmod m$

O(1)

BIG O

SET key1 “Hello”

T(h(“key1”)) = “Hello”

O(1)

n = len(params) / 2

MSET key1 “Hello” key2 “World”

```
for (i = 0; i < n; ++i) {
  T(h(params[i])) = params[i+1]
}
```

O(n)

time

n!

2^n

n^2

n log n

n

log n

1

n

O(1)

```
int add(int a, int b) {
  return a + b;
}
```

O(n)

```
int search(int a[], int n, int x) {
  for (int i = 0; i < n; ++i) {
    if (a[i] == x) {
      return i;
    }
  }
  return -1;
}
```

O(log n)

```
int binarySearch(int a[], int start, int end, int x) {
  if (start > end) return -1;
  int p = start + (end - start) / 2;
  if (a[p] == x) return p;
  if (x < a[p]) return binarySearch(a, start, p - 1, x);
  return binarySearch(a, p + 1, end, x);
}
```

O(n^2)

```
void insertionSort(int a[], int n) {
  for (int i = 0; i < n; ++i) {
    for (int j = i + 1; j < n; ++j) {
      if (a[i] > a[j]) {
        int t = a[i];
        a[i] = a[j];
        a[j] = t;
      }
    }
  }
}
```

LPUSH key value [value ...]

Available since 1.0.0.

Time complexity: O(1)

Insert all the specified values at the head of the list stored at `key`. If `key` does not exist, it is created as empty list before performing the push operations. When `key` holds a value that is not a list, an error is returned.

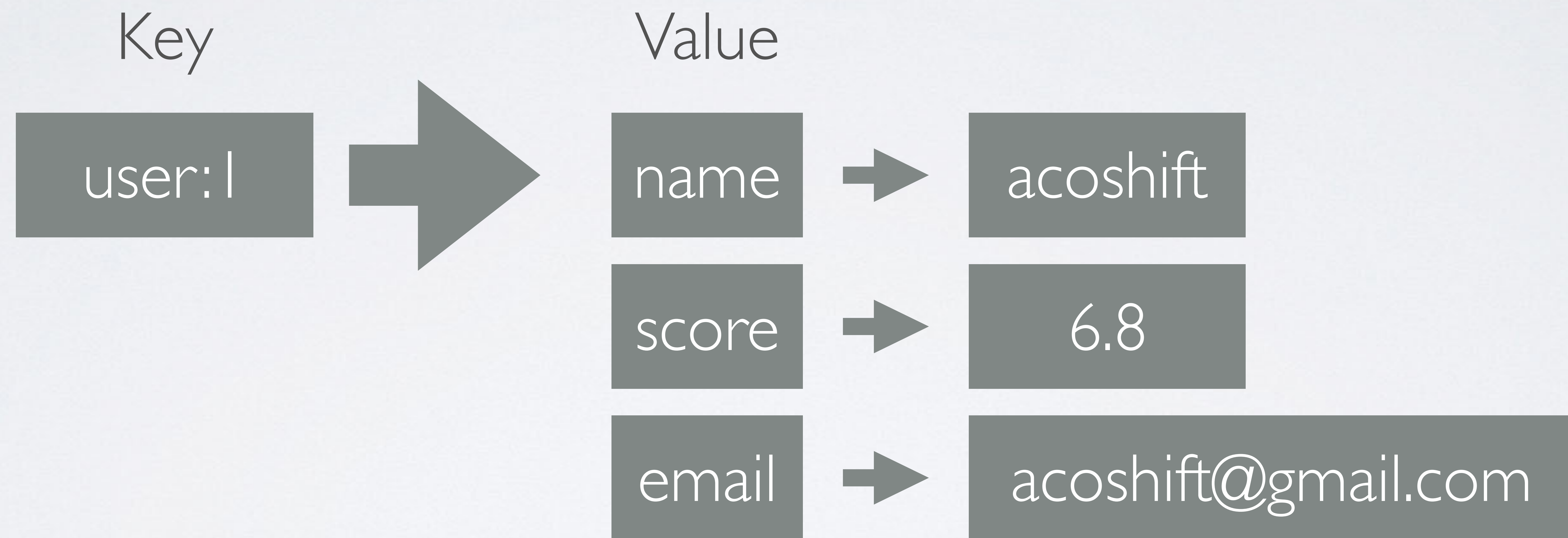
It is possible to push multiple elements using a single command call just specifying multiple arguments at the end of the command. Elements are inserted one after the other to the head of the list, from the leftmost element to the rightmost element. So for instance the command `LPUSH mylist a b c` will result into a list containing `c` as first element, `b` as second element and `a` as third element.

Return value

Integer reply: the length of the list after the push operations.

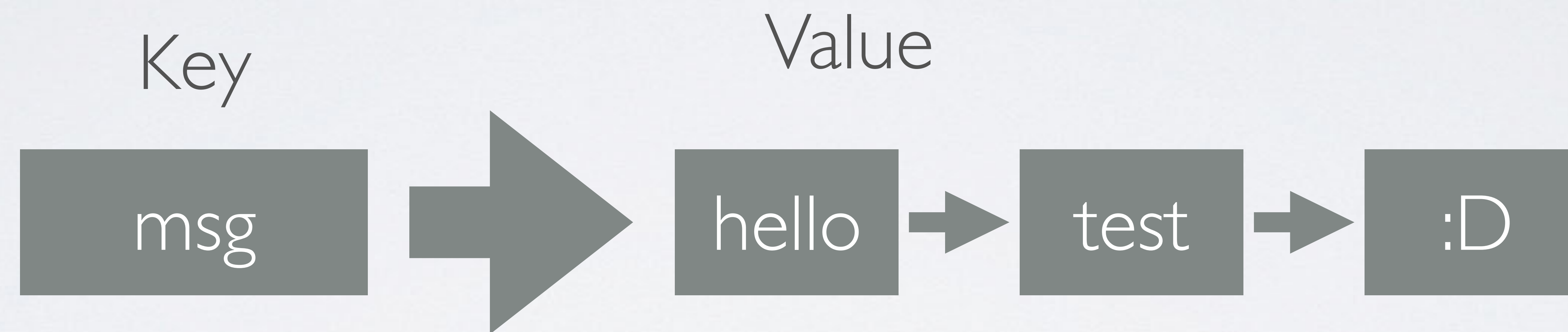
HASH

(Unordered hash table of keys to values)



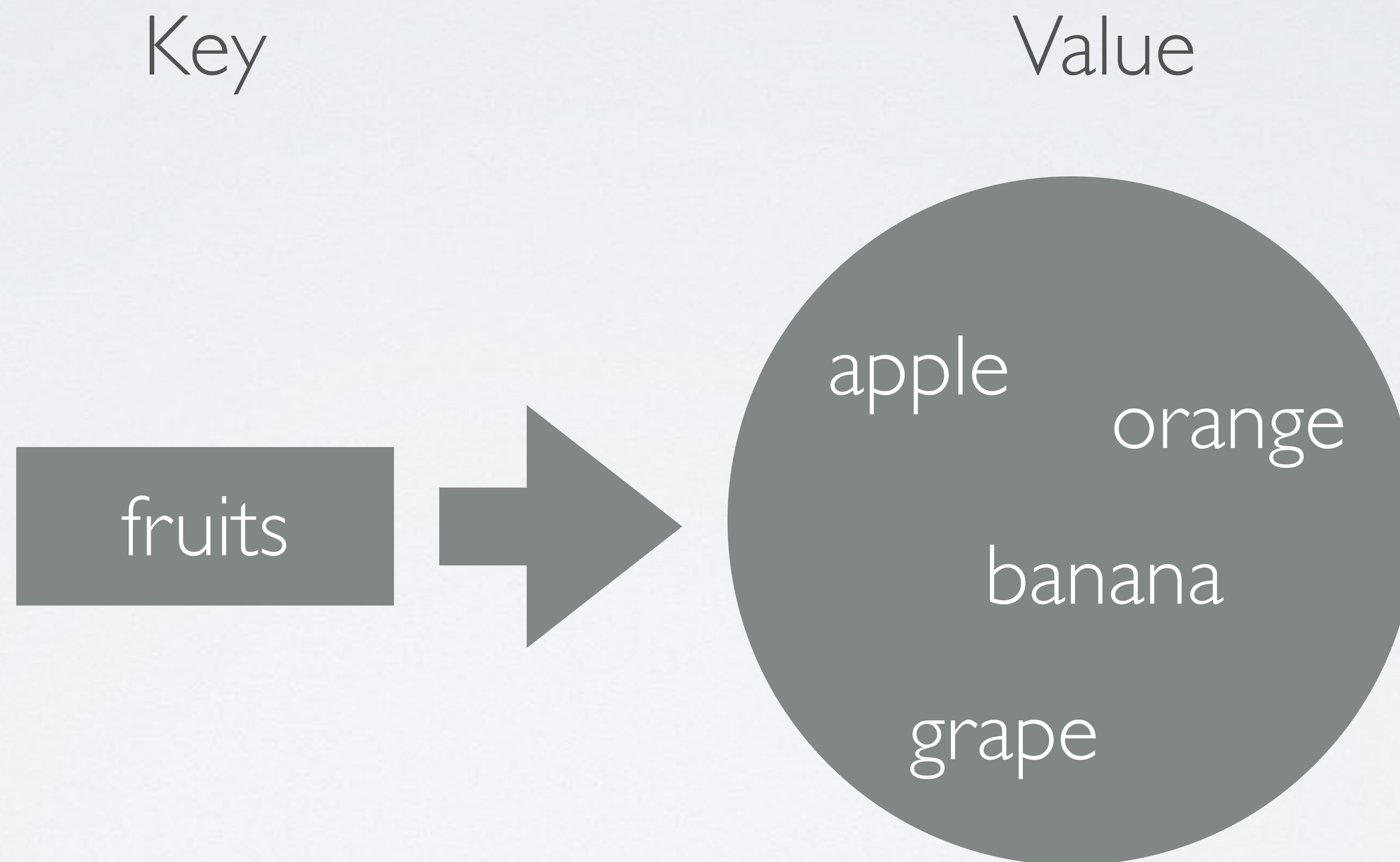
LIST

(Linked-list of Strings)



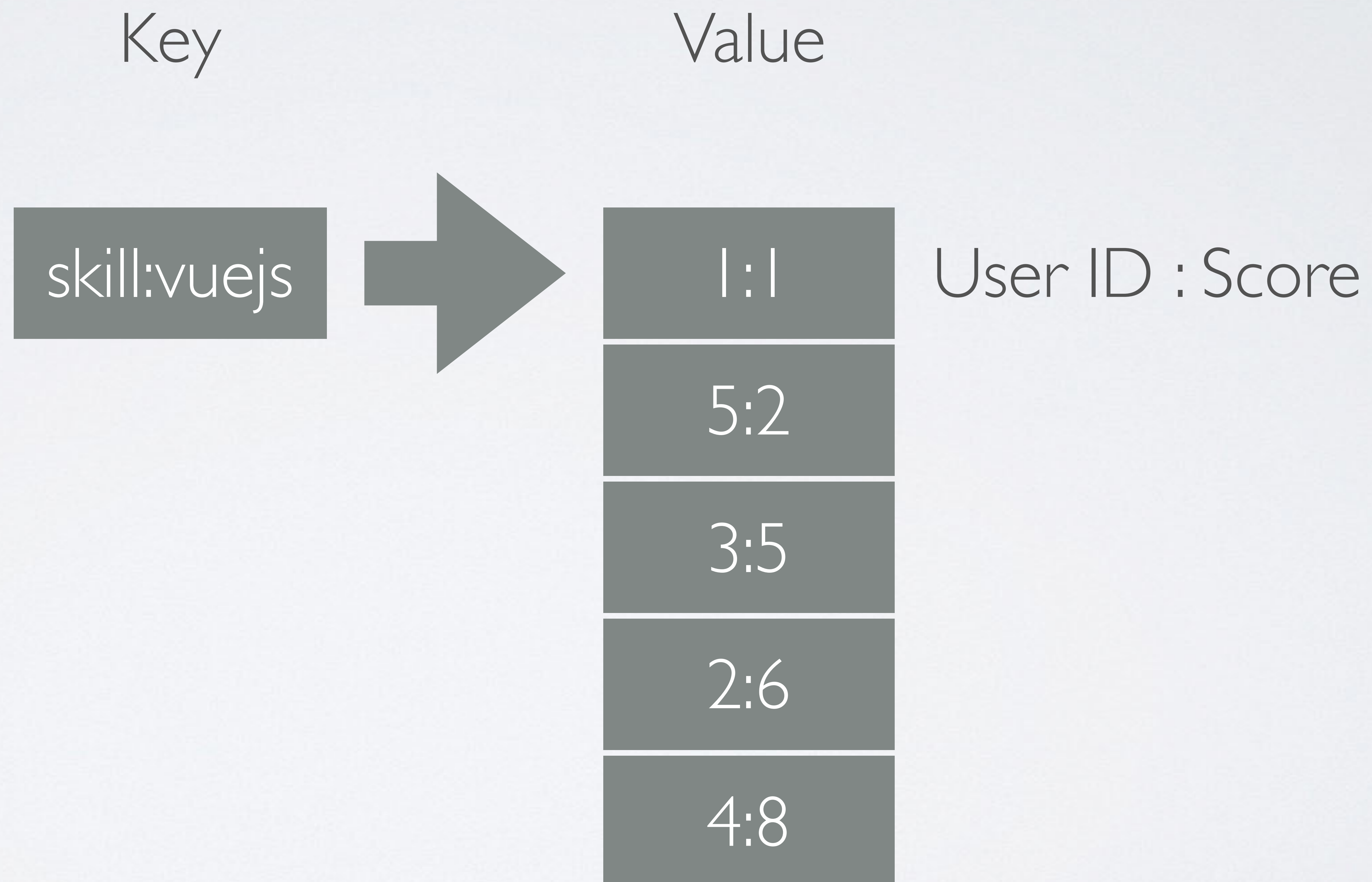
SET

(<u>Unordered</u> collection of <u>unique</u> Strings)



SORTED SET

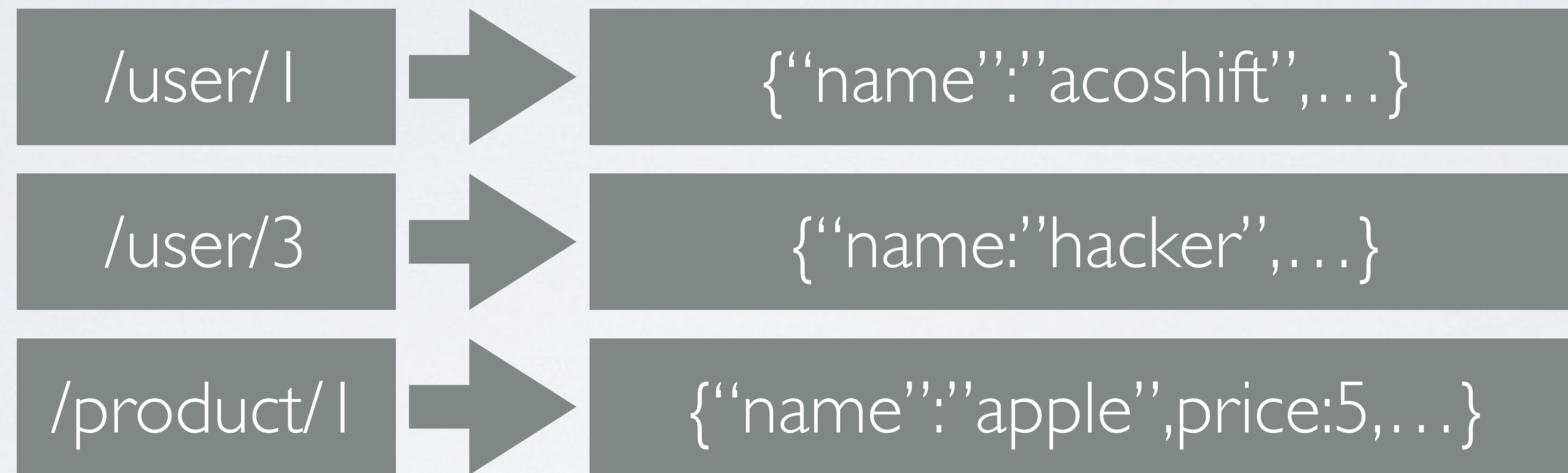
(Ordered mapping of string members to floating-point scores, ordered by score)



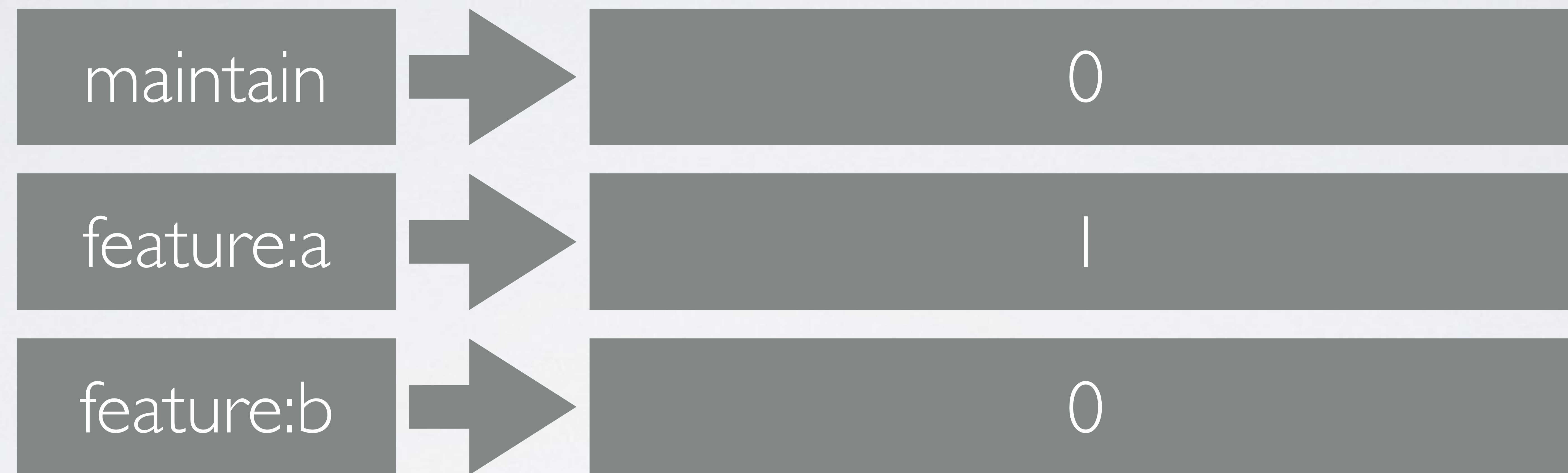
USE-CASES

LOOKUP

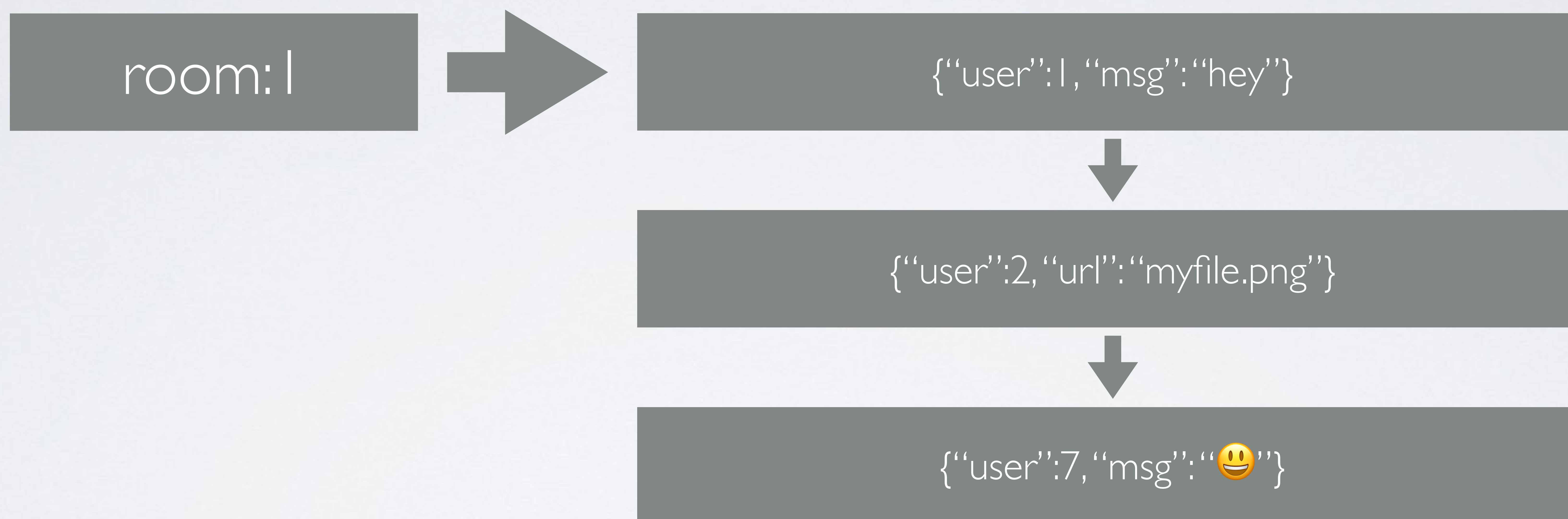
CACHE



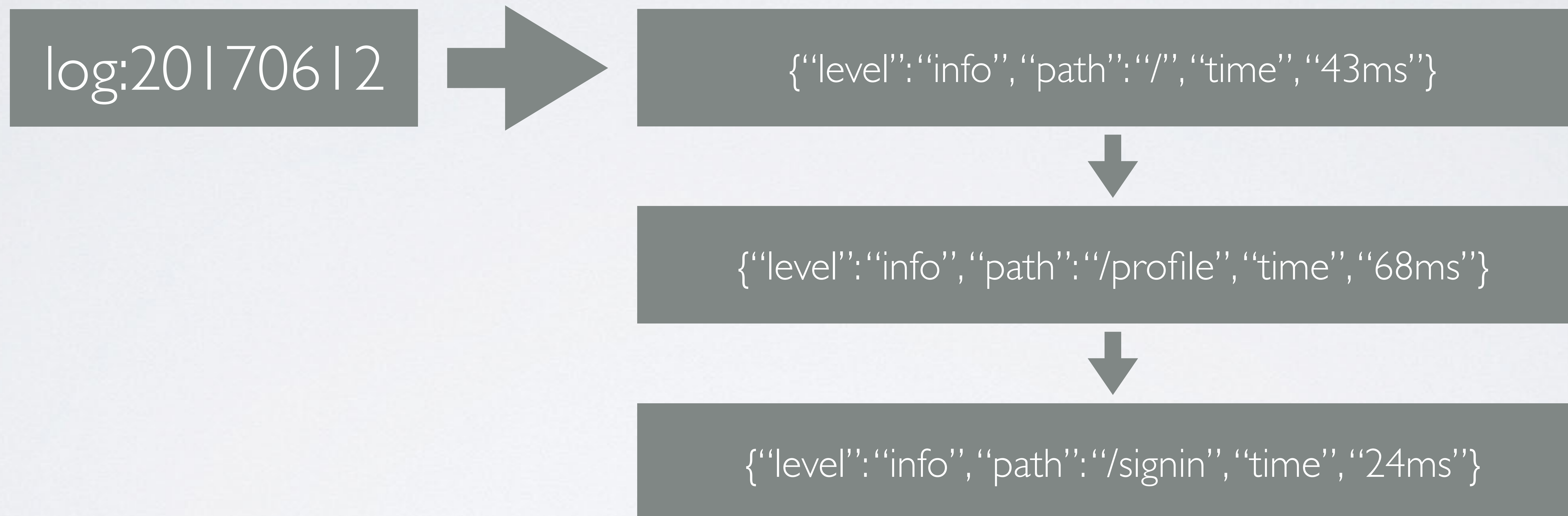
CONFIG



MESSAGE / COMMENT

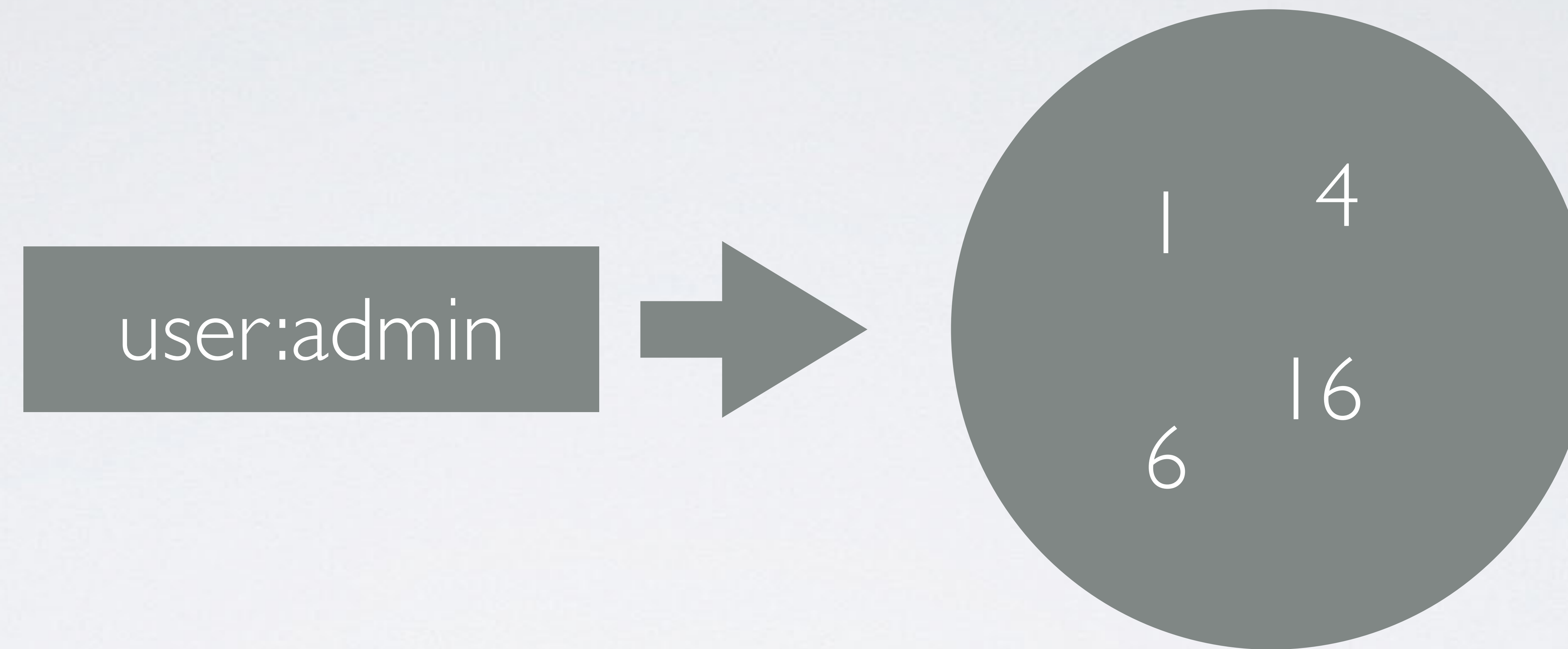


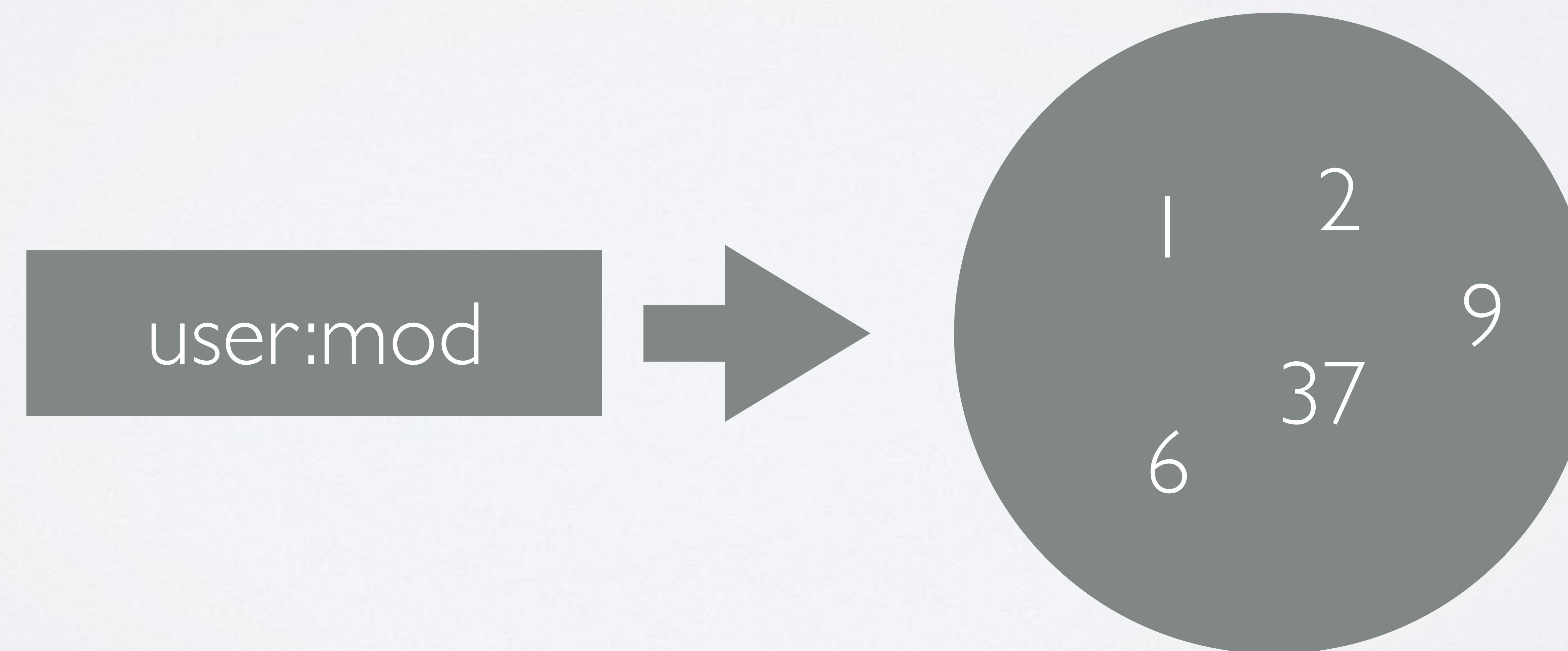
LOGGING



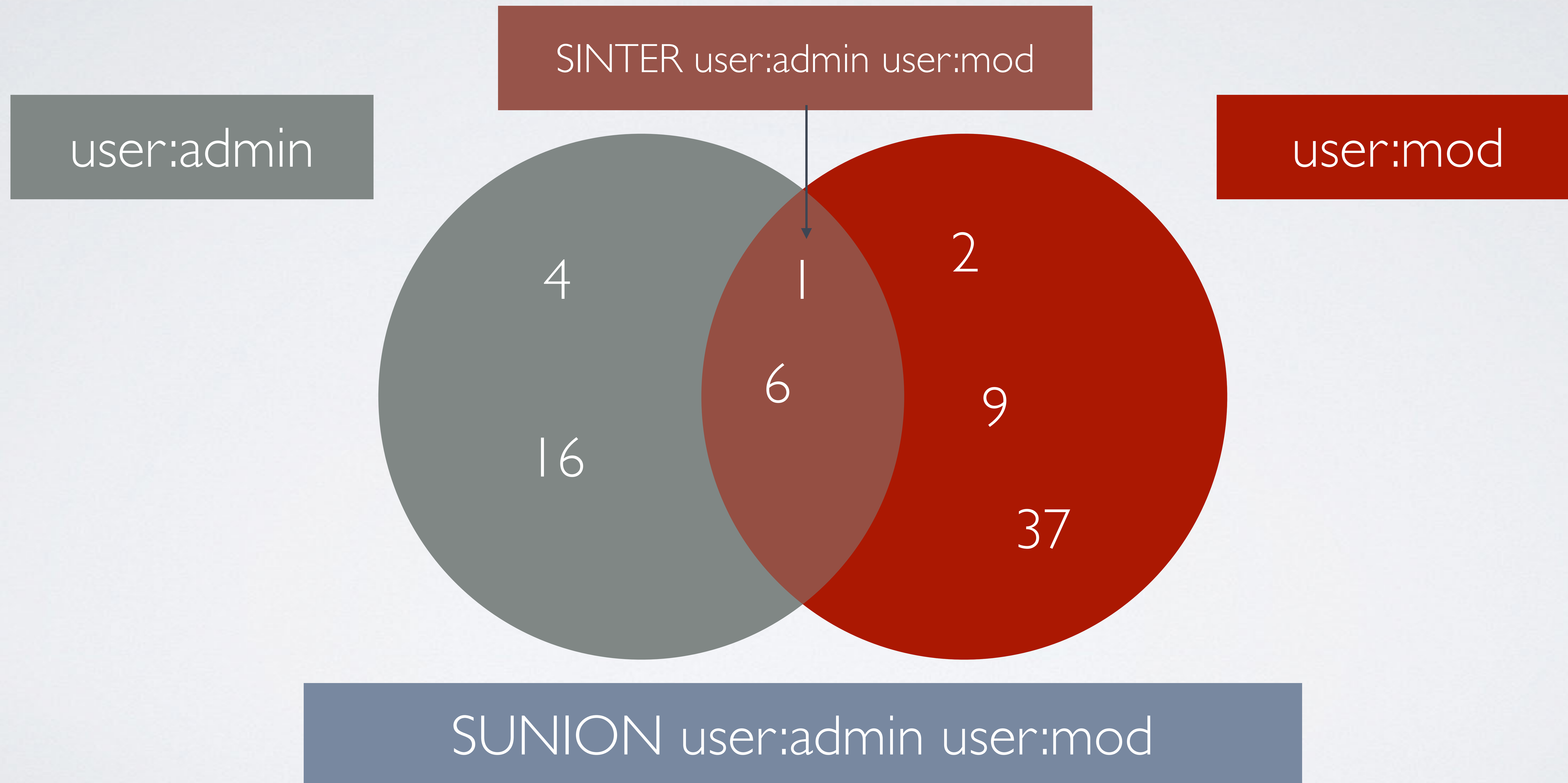
NOTIFICATION

ROLE





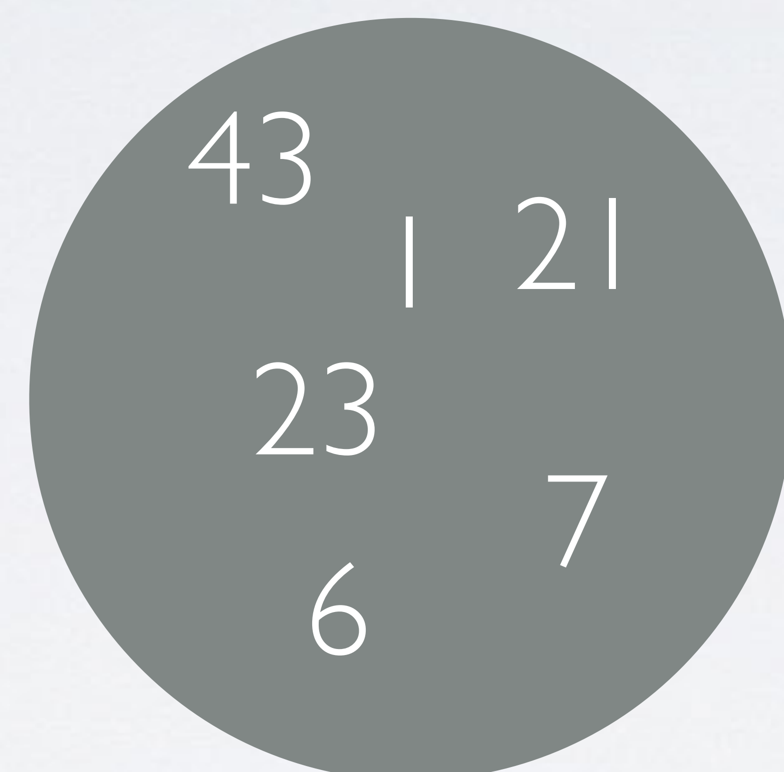
ROLE



TAG

tag:animal_ears

tag:seifuku

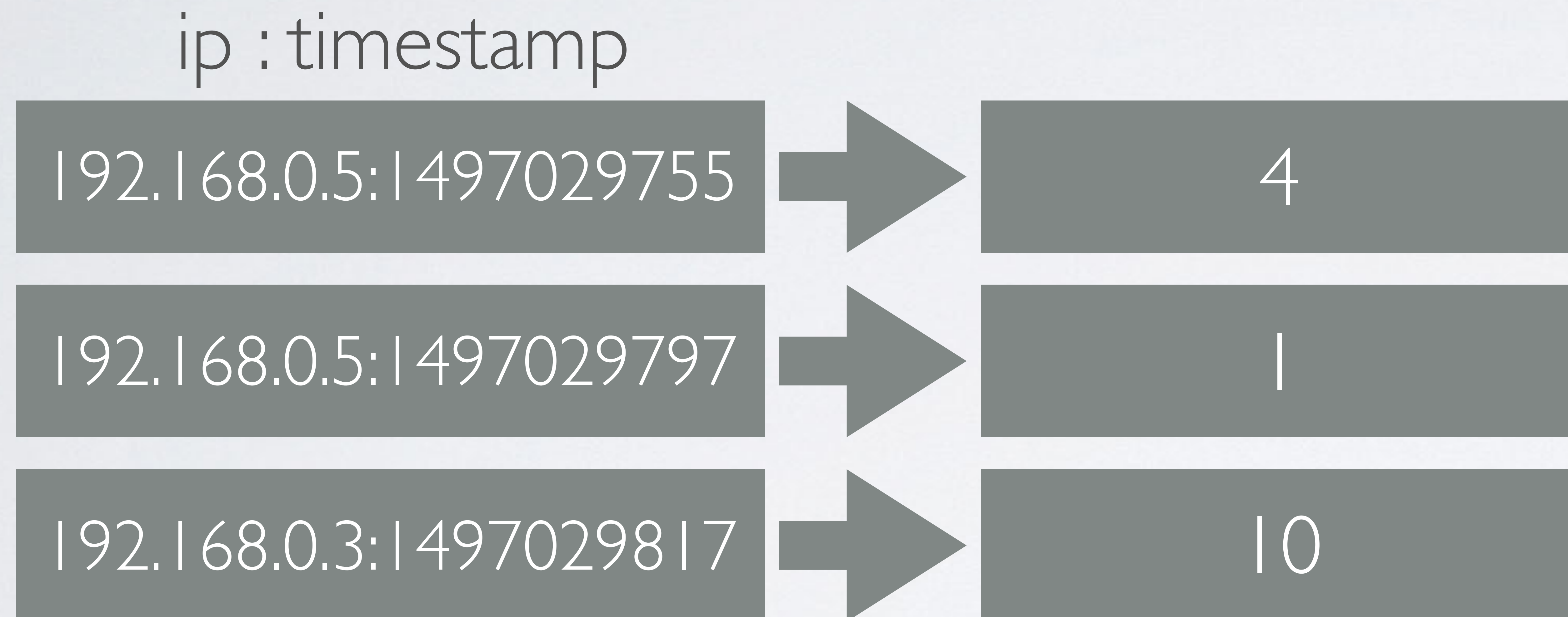


tag:kantoku

type:general

type:artist

RATE LIMITER




```
key = ip + “:” + ts // ts is timestamp
// roundup to second
current = GET(key)
if current > 10 { // max request = 10
    error “too many request per second”
} else {
    MULTI
        INCR(key)
        EXPIRE(key, 1)
    EXEC
}
```

FOLLOWER

Follower Count:
SCARD(user:1:follower)

Is user 2 follow user 1:
SISMEMBER(user:1:follower, 2)

Users who follow user 1:
SMEMBERS(user:1:follower)

Random 5 users who follow user 1:
SRANDMEMBER(user:1:follower, 5)

RANKING

post:like

{1:14}

{2:56}

{3:1}

{4:1490}

{5:321}

Like: ZINCRBY post:like 1 5

Unlike: ZINCRBY post:like -1 5

LIST BY TIMESTAMP

post:all

1
2
3
5

post:created_at

{1:1497030475}
{2:1497030497}
{3:1497030517}
{4:1497030533}
{5:1497030585}

ZINTERSTORE
result
2
post:all
post:created_at
WEIGHTS
1
1
AGGREGATE MAX

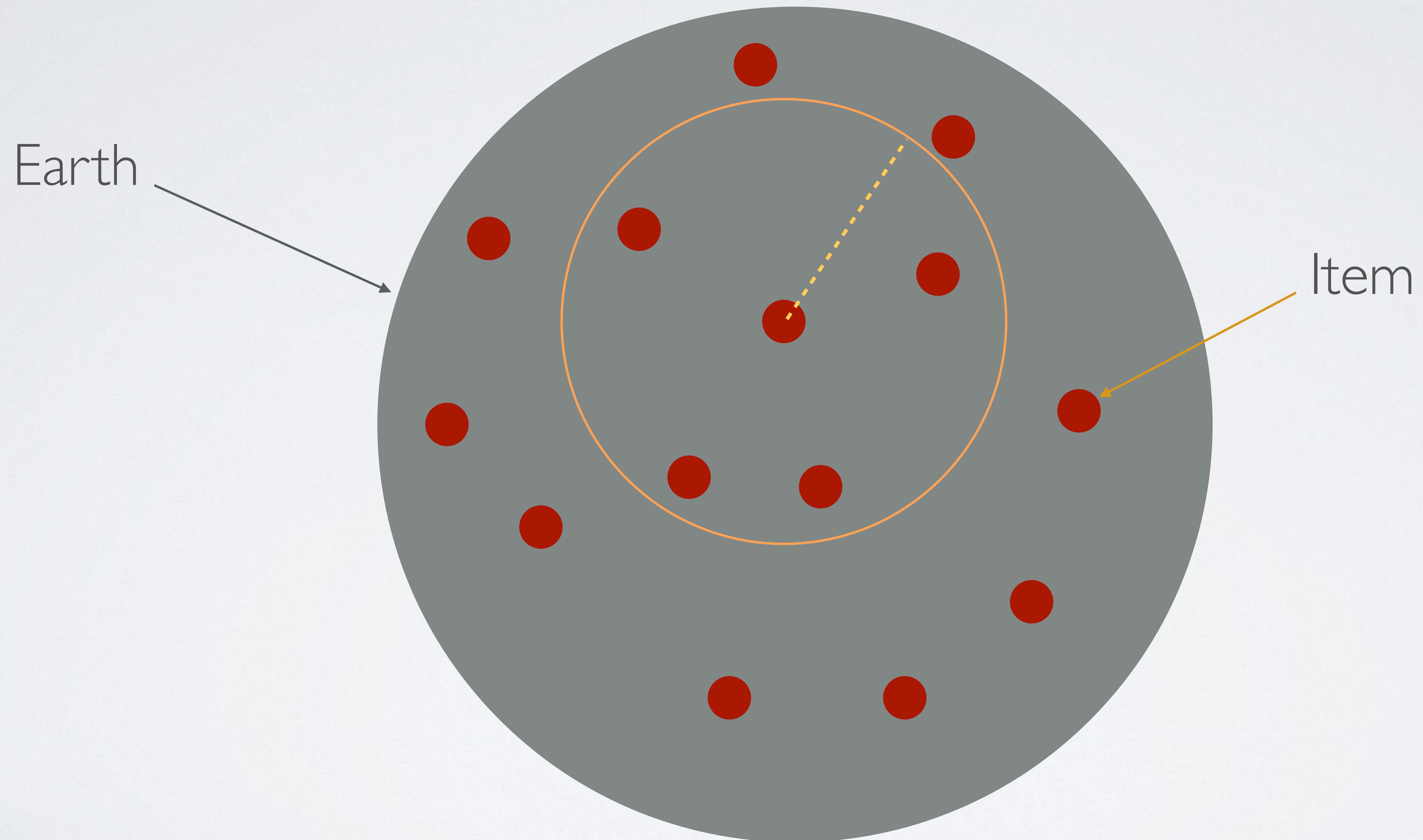
result

{1:1497030475}
{2:1497030497}
{3:1497030517}
{5:1497030585}

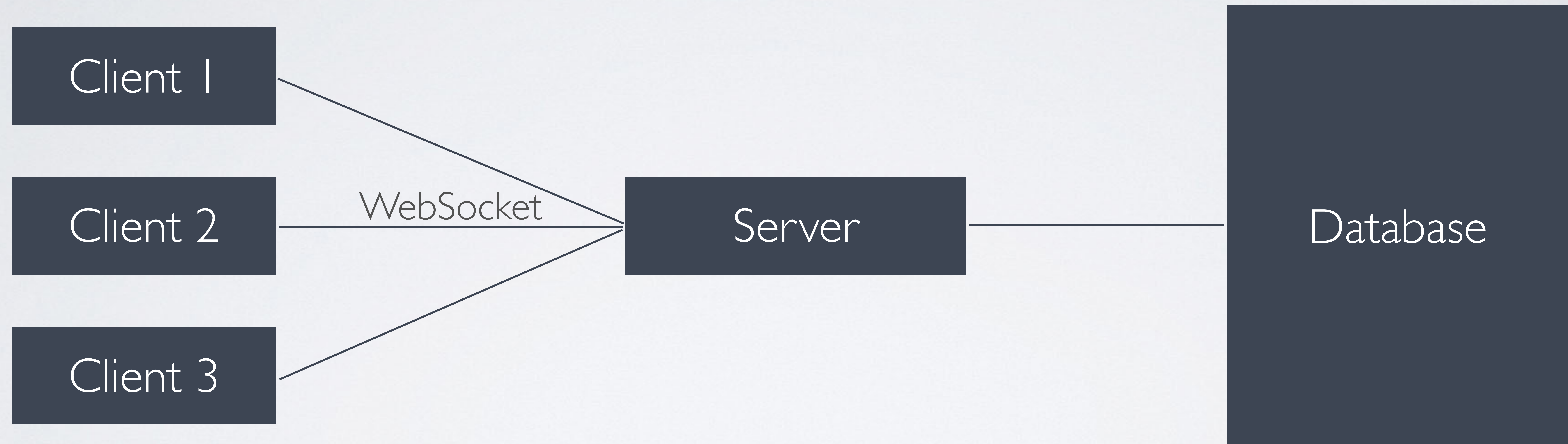
HYPERLOGLOG

(Probabilistic data structure, for count unique things)

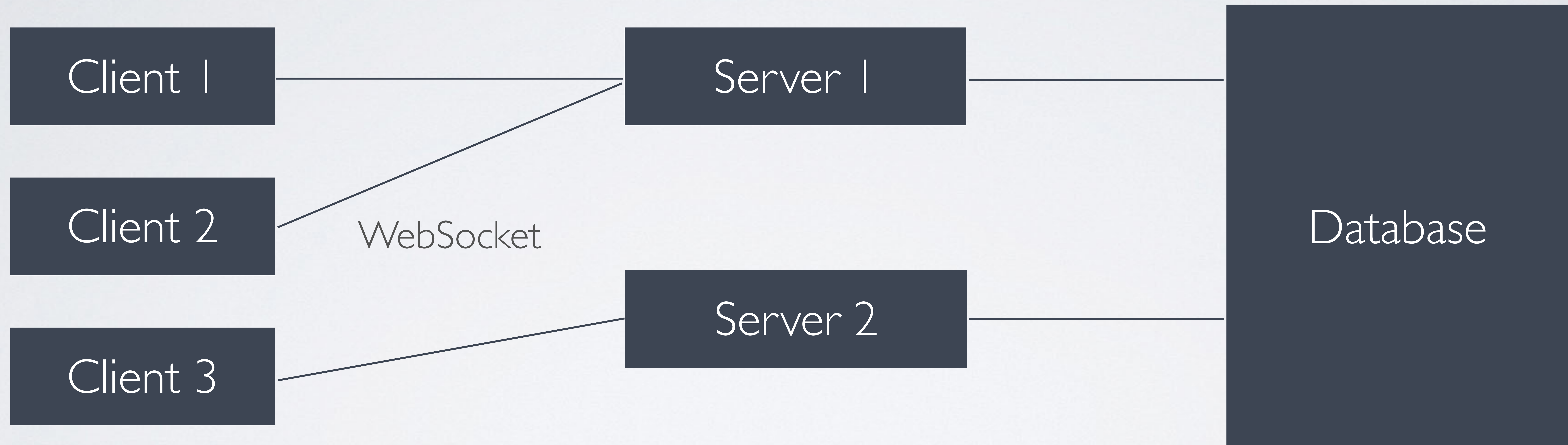
GEO



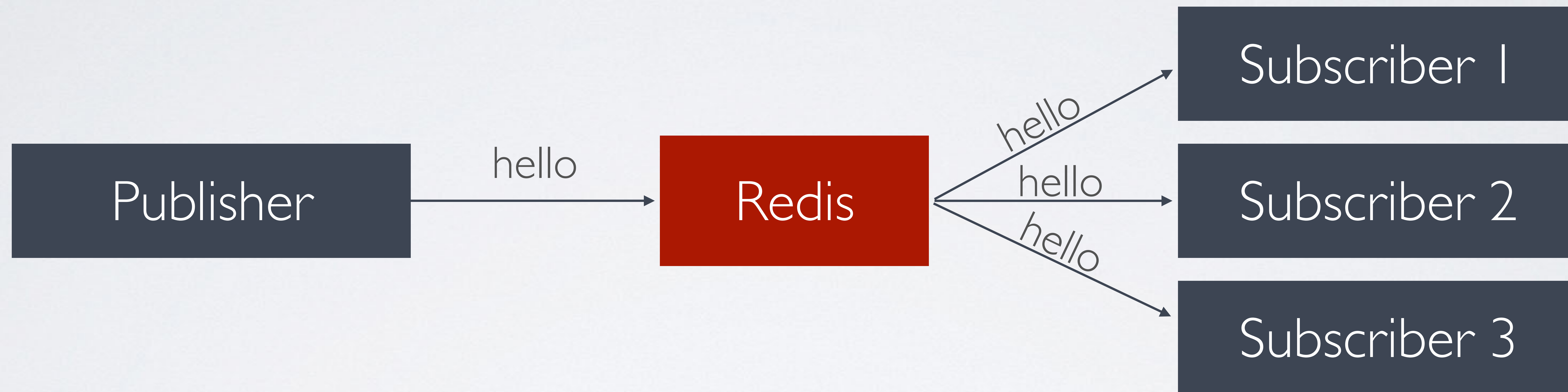
PUBSUB



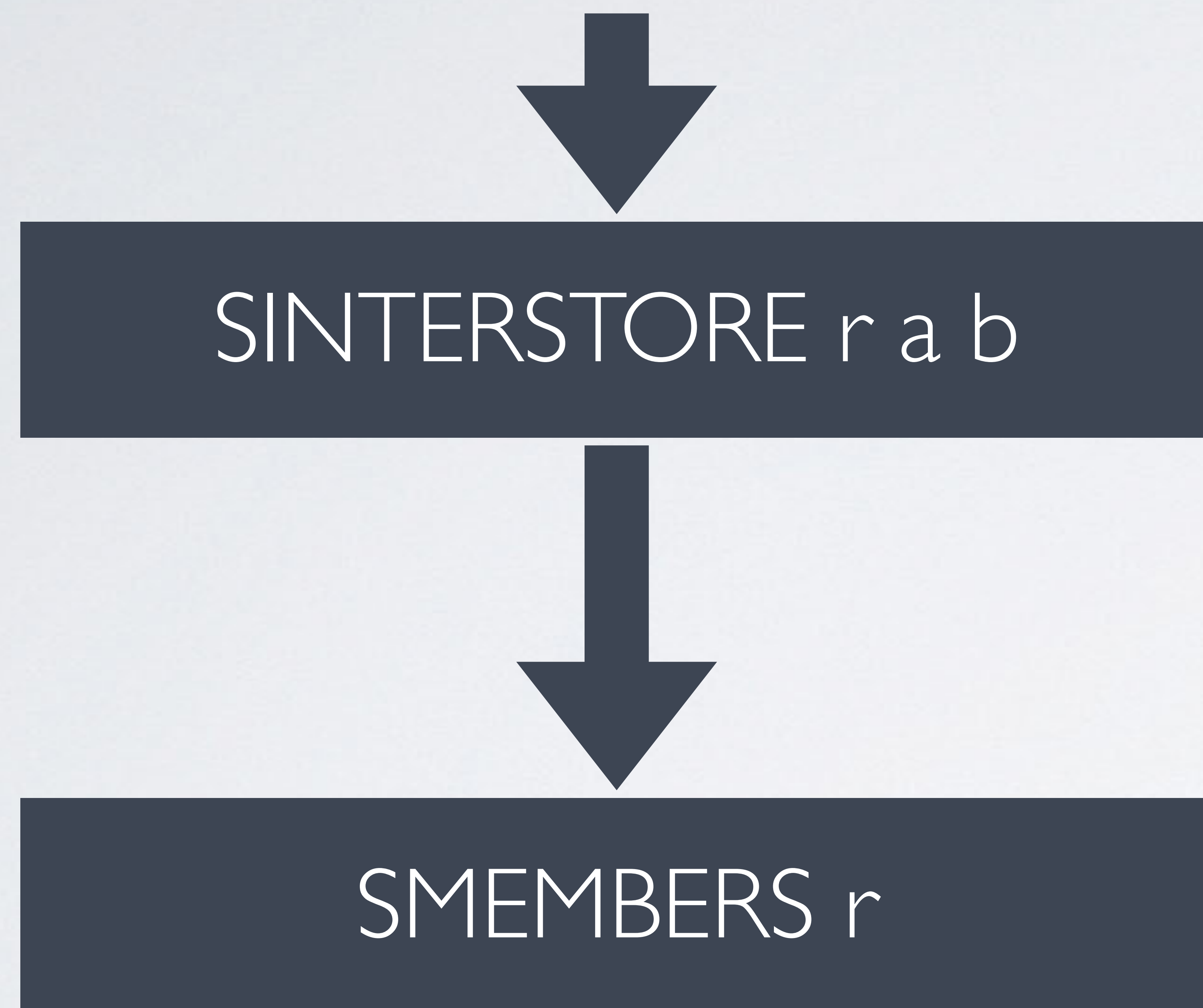
PUBSUB

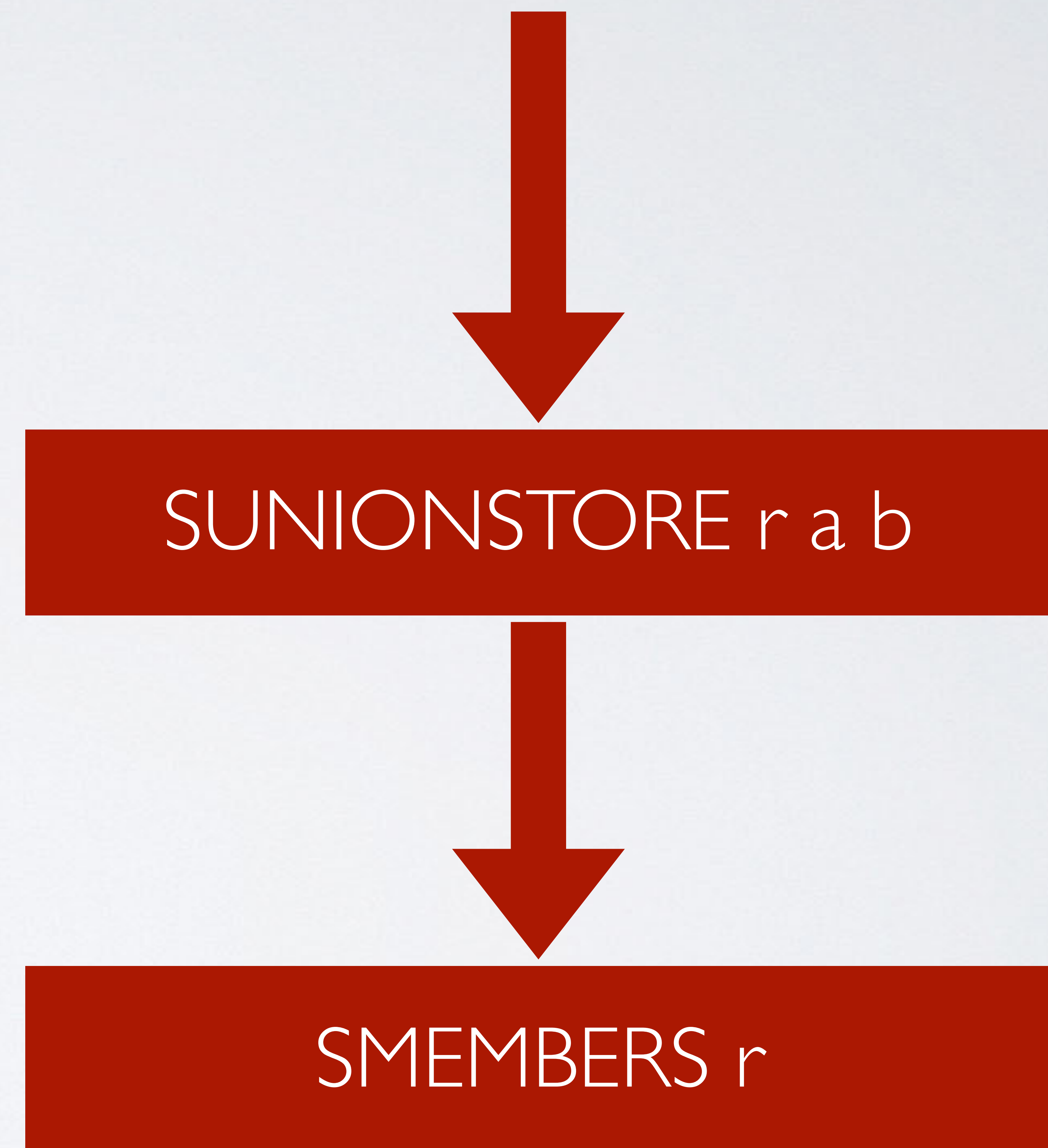


PUBSUB



TRANSACTION



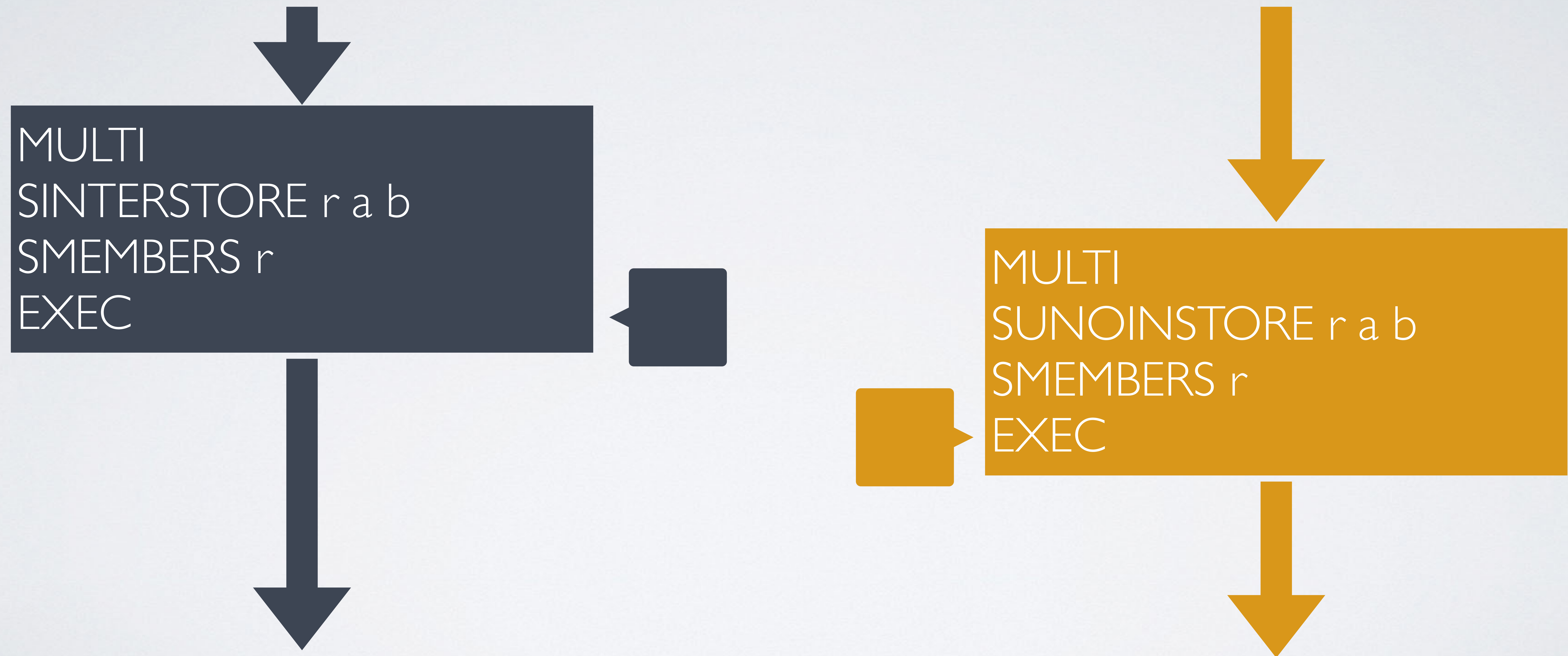


TRANSACTION

MULTI
SET a 1
SET b 2
EXEC

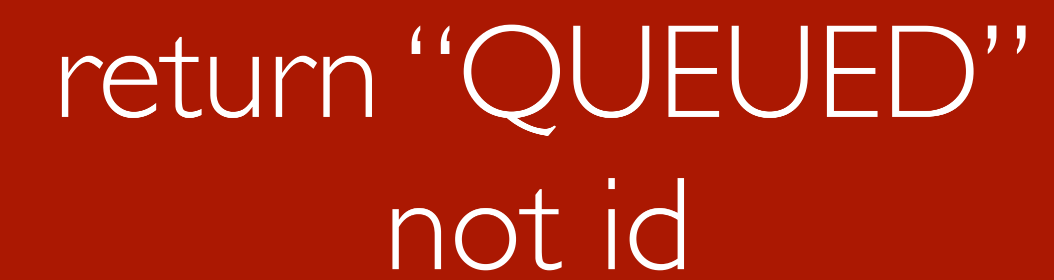
MULTI
SET a 1
SET b 2
DISCARD

TRANSACTION



TRANSACTION

MULTI
id := GET id
SET id (id + 1)
EXEC



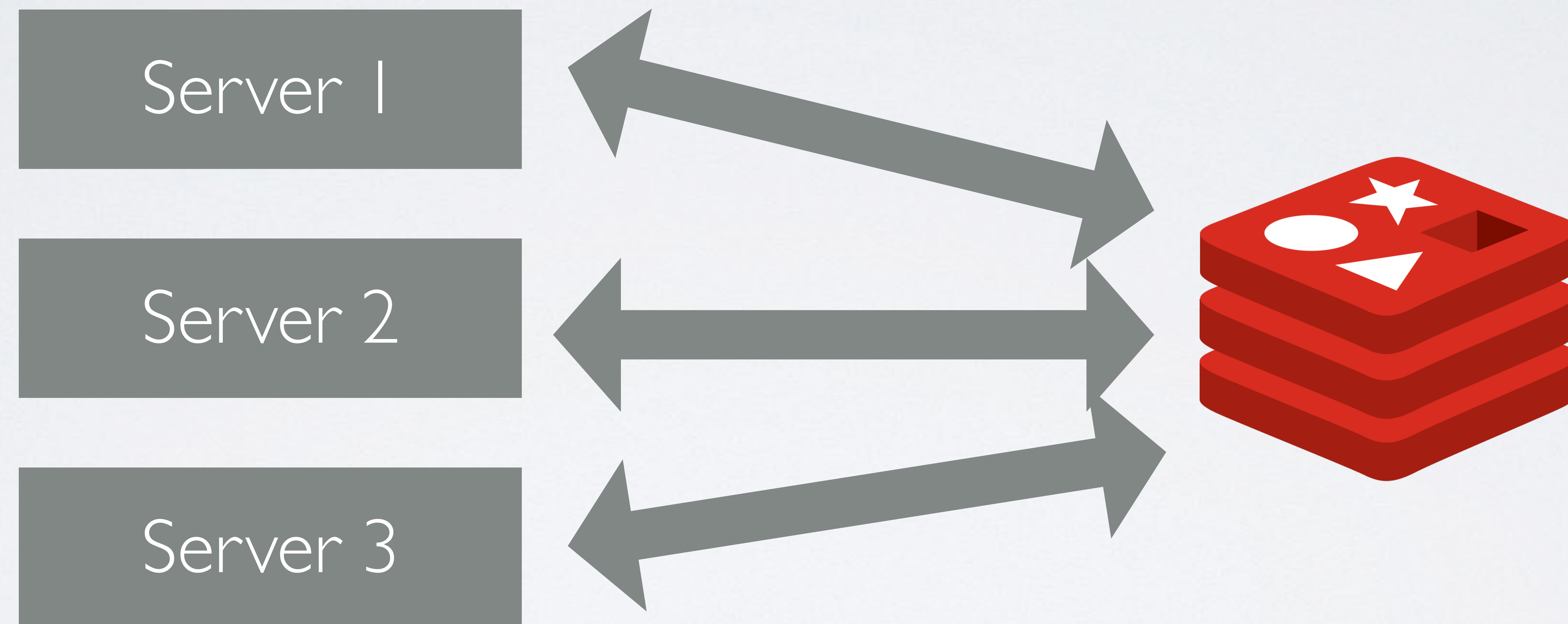
TRANSACTION

WATCH id
id := GET id
MULTI
SET id (id + 1)
EXEC

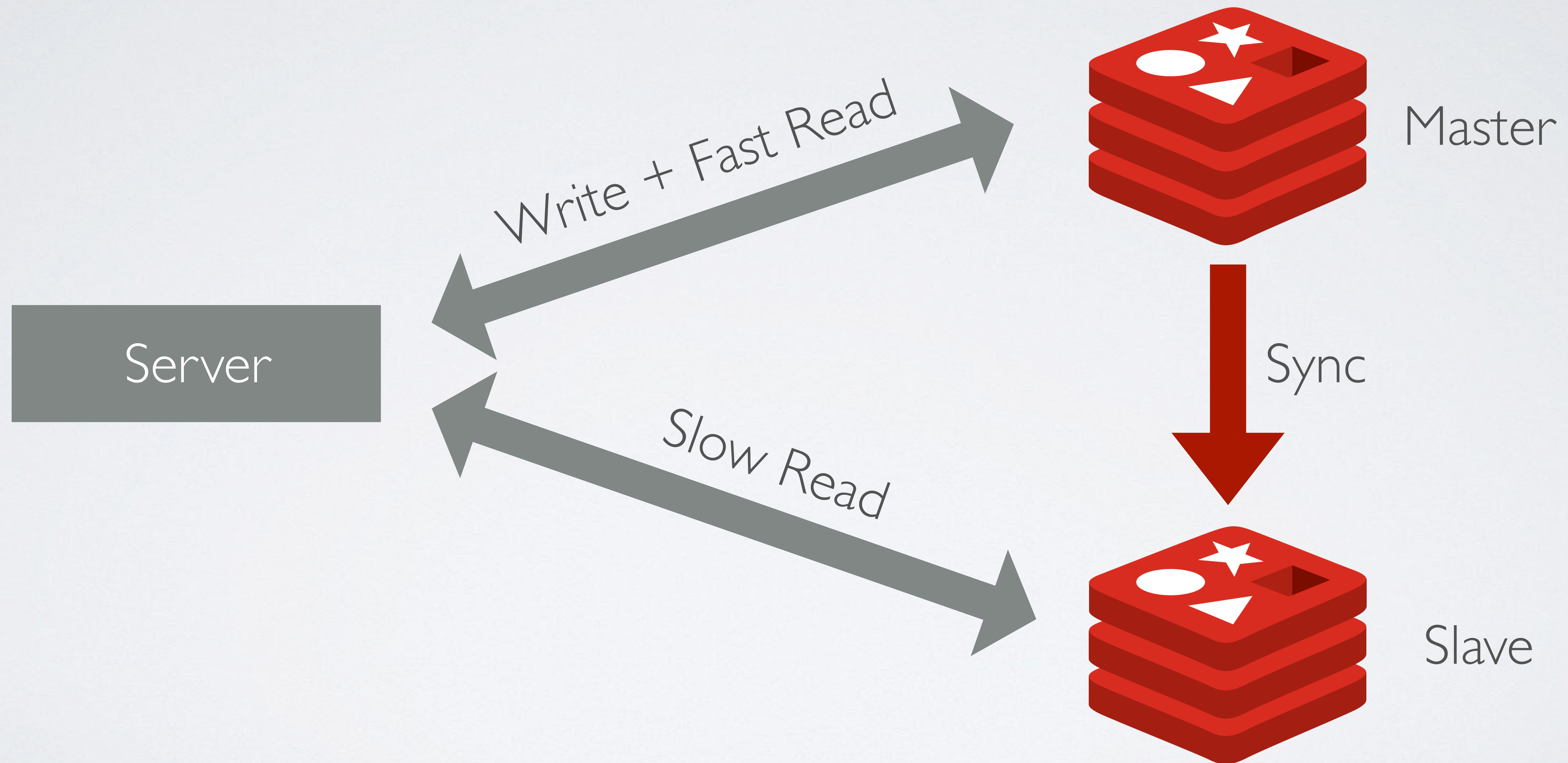
CONFIG REDIS

- http://download.redis.io/redis-stable/redis.conf
- Persistence
  - RDB (Snapshot)
  - AOF (Append-only File)

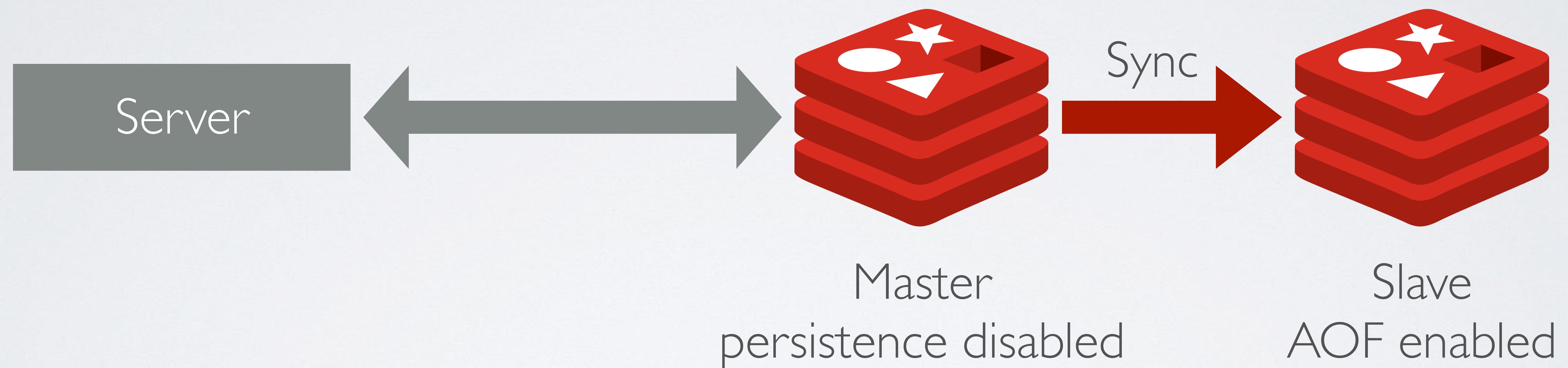
REPLICATION



REPLICATION



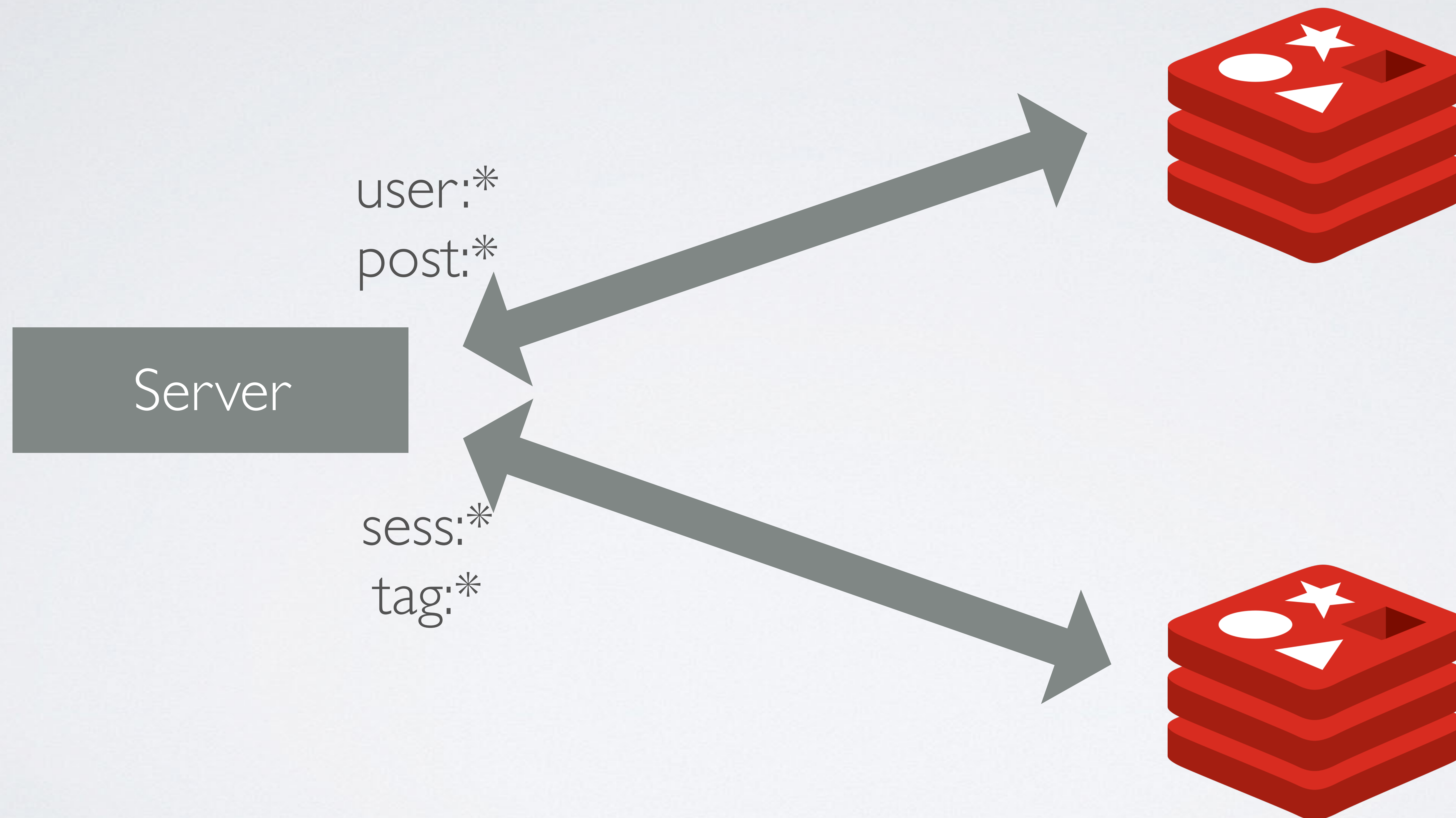
DISKLESS REPLICATION



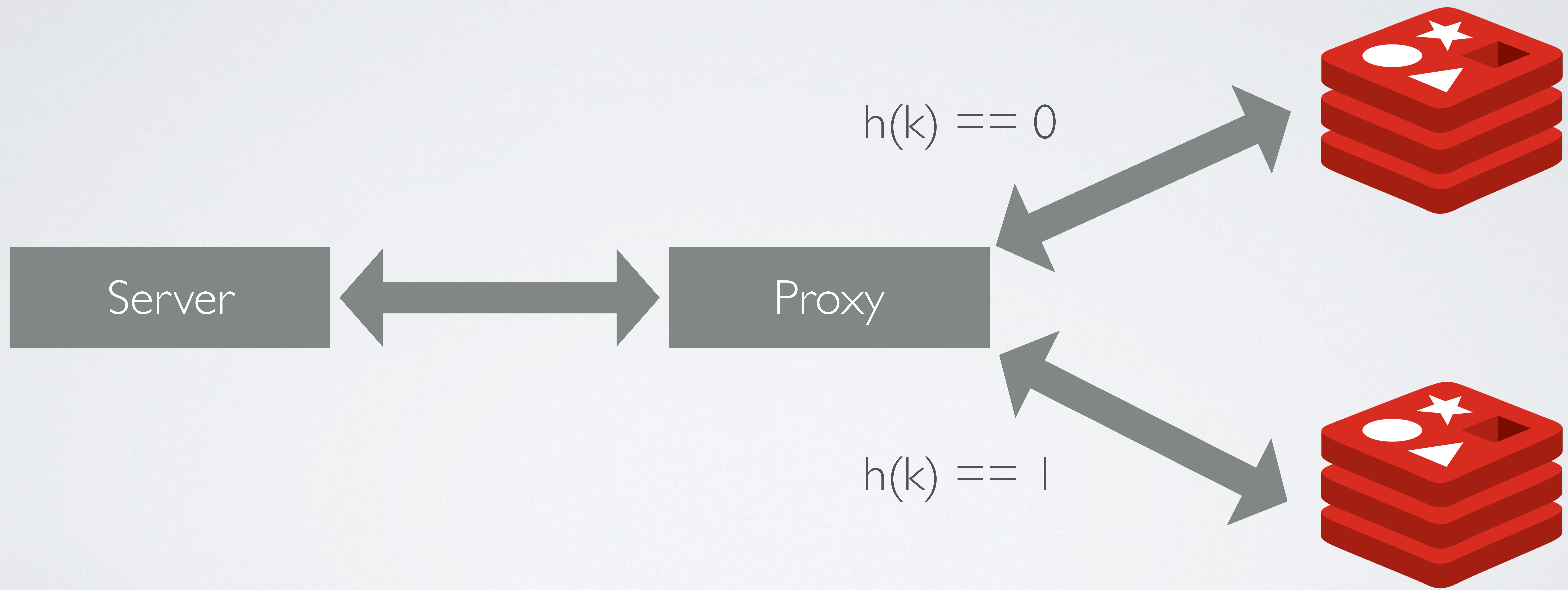
PARTITIONING

- Client side partitioning
- Proxy assisted partitioning
- Query routing (Redis Cluster)

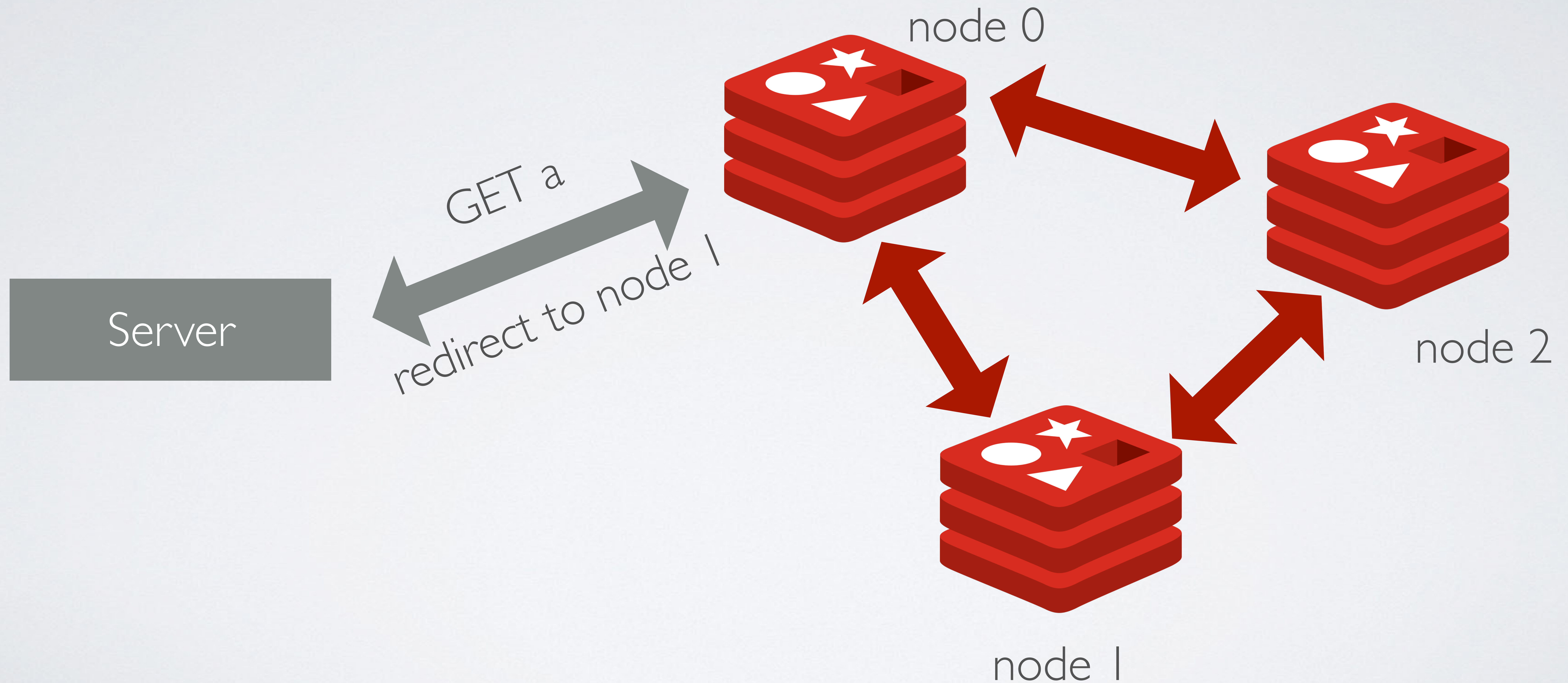
CLIENT SIDE



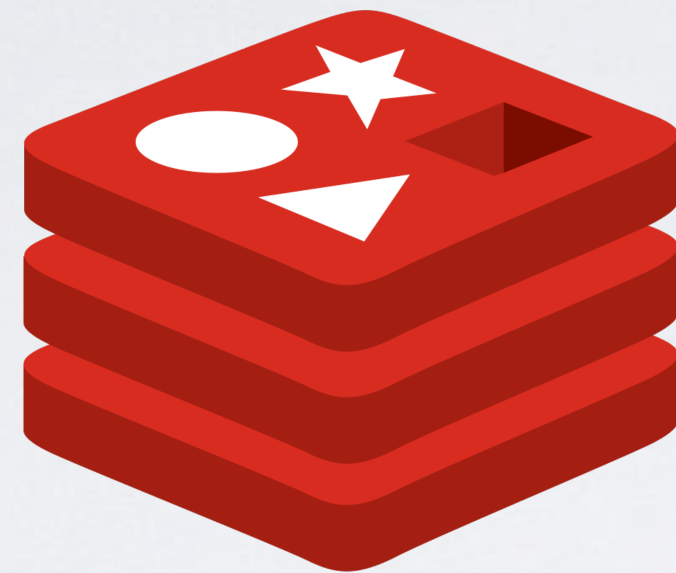
PROXY ASSISTED



QUERY ROUTING



REDIS CLUSTER



3 x (master + 1 slave)

```
$ gem install redis
$ ./redis-trib.rb create --replicas 1 127.0.0.1:7000 \
127.0.0.1:7001 127.0.0.1:7002 127.0.0.1:7003 \
127.0.0.1:7004 127.0.0.1:7005
$ redis-cli -c -p 7000
```

SCRIPTING

- EVAL — Run script
- SCRIPT LOAD — Load script into cache
- EVALSHA — Run script from cached

EVAL script numkeys key [key ...] arg [arg …]

```
eval "return {KEYS[1],KEYS[2],ARGV[1],ARGV[2],ARGV[3]}"
2 key1 key2 arg1 arg2 arg3
1) "key1"
2) "key2"
3) "arg1"
4) "arg2"
5) "arg3"
```

DATABASE DESIGN

Q&A